Vysoká škola báňská – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Dvoufaktorová autentizace ve virtuálních privátních sítích

Dual-factor Authentication in Virtual Private Networks

2015 **Bc. Michal Tabaček**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání diplomové práce

Student: **Bc. Michal Tabaček**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T059 Mobilní technologie

Téma: Dvoufaktorová autentizace ve virtuálních privátních sítích
Dual-factor Authentication in Virtual Private Networks

Zásady pro vypracování:

Cílem diplomové práce je návrh, realizace a otestování různých řešení virtuálních privátních sítí, které využívají dvoufaktorovou autentizaci (USB token a heslo).

Osnova práce:
1. Popište princip vícefaktorové autentizace a možnosti jejího použití ve virtuálních privátních sítích, které jsou založeny na softwarech OpenVPN a strongSwan.
2. Navrhněte a v laboratorních podmínkách realizujte různé druhy sítí VPN, které využívají dvoufaktorovou autentizaci. Ověřte funkčnost navržených řešení.
3. Srovnejte jednotlivá řešení a zhodnoťte jejich výhody a nevýhody.

Seznam doporučené odborné literatury:

CARMOUCHE, James Henry. *IPsec Virtual Private Network Fundamentals*. Indianapolis: Cisco Press, 2006. ISBN 1-58705-207-5.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Petr Machník, Ph.D.**

Datum zadání: 01.09.2014
Datum odevzdání: 07.05.2015

doc. Ing. Miroslav Vozňák, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne: *4. května 2015*

……..……
podpis studenta

Poděkování

Rád bych poděkoval Ing. Petru Machníkovi, Ph.D. za odbornou pomoc a vstřícné konzultace při vytváření této diplomové práce.

Abstrakt

Cílem této diplomové práce je návrh a realizace dvoufaktorové autentizace ve virtuálních privátních sítích pomocí USB tokenu a hesla. Pro praktickou realizaci navržených řešení je použit software OpenVPN a strongSwan. Nachází se zde kompletní návod na instalaci a práci s USB tokenem. K vytvoření a práci s certifikáty je využit nástroj Easy RSA a software XCA. U navržených řešení jsou uvedeny jednotlivé konfigurace a konfigurační soubory. Dále je popsáno ověření funkčnosti dvoufaktorové autentizace a připojení klientů z operačních systémů Ubuntu a Windows. Ověření funkčnosti je doplněno o výpisy sestavených spojení. Na závěr jsou mezi sebou jednotlivá řešení srovnána.

Klíčová slova

Autentizace, certifikát, dvoufaktorová autentizace, EAP, IPsec, konfigurace, OpenVPN, parametr, privátní klíč, RADIUS, síť, SSL/TLS, strongSwan, Ubuntu, USB token, VPN.

Abstract

The goal of this master thesis is proposal and realization of dual-factor authentication in virtual private networks using USB token and password. Practical realization of proposed solutions is going to be made using OpenVPN and strongSwan software. Complete instructions for installation and operation of USB token is described here. Easy RSA tool and XCA software are used to create the certificates. Proposed solutions are listed with configurations and configuration files. They are followed by a description of verification of functionality of dual-factor authentication and connection of clients from Ubuntu and Windows operating systems. Verification is accompanied by listing of compiled connections. In the end each solutions are compared.

Key words

Authentication, certificate, dual-factor authentication, EAP, IPsec, configuration, OpenVPN, parameter, private key, RADIUS, network, SSL/TLS, strongSwan, Ubuntu, USB token, VPN.

Seznam použitých zkratek

Zkratka	Význam
3DES	3 (Triple) Data Encryption Standard
AES	Advanced Encryption Standard
AH	Authentication Header
API	Application Programming Interface
CA	Certificate Authority
CCID	Chip/Smart Card Interface Devices
CN	Common Name
CRL	Certificate Revocation List
DDL	Data Definition Language
DES	Data Encryption Standard
D-H	Diffie-Hellman
DoS	Denial of Service
EAP	Extensible Authentication Protocol
ECDSA	Elliptic Curve Digital Signature Algorithm
EKU	Extended Key Usage
ESP	Encapsulating Security Payload
GCC	GNU Compiler Collection
GNU GPL	GNU General Public License
GRE	Generic Routing Encapsulation
GUI	Graphical User Interface
HMAC	Keyed-hash Message Authentication Code
IANA	Internet Assigned Numbers Authority
IKE	Internet Key Exchange
IP	Internet Protocol
IPsec	Internet Protocol Security
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6

ISAKMP	Internet Security Association and Key Management Protocol
ISO/OSI	International Standards Organization / Open Systen Interconnection
ISP	Internet service provider
L2TP	Layer 2 Tunneling Protocol
LAN	Local Area Network
MAC	Message Authentication Code
MD5	Message-Digest algorithm 5
MSCHAPV2	Microsoft Challenge-Handshake Authentication Protocol
MTU	Maximum transmission unit
NAT	Network Address Translation
OS	Operating system
PC	Personal Computer
PC/SC	Personal Computer/Smart Card
PEAP	Protected Extensible Authentication Protocol
PIN	Personal Identification Number
PKCS	Public Key Cryptographic Standards
PKI	Public Key Infrastructure
PPP	Point-to-Point Protocol
PPTP	Point-to-Point Tunneling Protocol
PSK	Pre-shared key
PUK	Personal Unblocking Key
RADIUS	Remote Authentication Dial In User Service
RSA	Rivest-Shamir-Adleman
SA	Security Association
SHA	Secure Hash Algorithm
SMS	Short message service
SOPIN	Security Officer Personal Identification Number
SQL	Structured Query Language
SSH	Secure Shell
SSL	Secure Sockets Layer

TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
USB	Universal Serial Bus
VPN	Time Division Multiplexing
WAN	Wide Area Network
Xauth	Extended Authentication
XCA	X Certificate and key management

Obsah

Úvod

V dnešním světě internetu a bezpečnostních hrozeb si můžeme jen těžko představit, že by veškerá komunikace v tomto světě probíhala nezabezpečeně. Pro ochranu naší komunikace a našich dat se využívájí různé technologie, šifrování, protokoly a metody autentizace. Jedna z možností jak zabezpečit naše data je vytvoření virtuálních privátních sítí (VPN). Virtuální síť VPN je v podstatě zabezpečený tunel, který je pomocí šifrovacích a autentizačních technologií zaveden nad existující již známou síťovou infrastrukturou.

VPN tunel je potřebné vytvořit hlavně v případě, kdy komunikace a data jsou citlivá na bezpečnost a nesmí se dostat do jiných rukou. Častým případem je právě připojení vzdálených pracovníků do firemní sítě. Zde narážíme na problém, že autentizace neboli proces ověřování identity je většinou závislý pouze na jednom faktoru (heslo nebo certifikát). Proto je dobré zavést vícefaktorovou autentizaci, kdy kombinujeme různé metody a přidáváme k autentizaci další hardwarové prvky (USB token, čipová karta atd.).

Diplomová práce, jak již název napovídá, pojednává o dvoufaktorové autentizaci ve virtuálních privátních sítích postavených na řešeních OpenVPN a strongSwan. V průběhu studia jsem se v několika předmětech okrajově setkal s OpenVPN. Řešení VPN mě zaujalo a chtěl jsem poznat další možnosti, proto jsem si zvolil toto téma k mé diplomové práci.

V první části popíši virtuální privátní sítě - jejich výhody, nevýhody a důvody zavedení. Dále popisuji požadavky kladené na VPN, jejich rozdělení dle typu spojení a použitého protokolu. Po seznámení s virtuálními privátními sítěmi je další kapitola věnována vícefaktorové autentizaci a popisu metod autentizace. V závěru této teoretické části je krátce popsán použitý software OpenVPN a strongSwan.

Nejdůležitější částí této diplomové práce je návrh, realizace a otestování vytvořené virtuální privátní sítě, která využívá dvoufaktorovou autentizaci (USB token a heslo). Jako první jsou krátce popsány certifikáty a certifikační autorita, následuje návod na instalaci a popis práce s USB tokenem několika způsoby. Testování bude probíhat v laboratorním prostředí. Tato druhá část obsahuje seznámení s použitým softwarem, postup instalace a kompletní postup konfigurace. Nakonec je pro dvoufaktorovou autentizaci ověřena funkčnost a jsou mezi sebou srovnány řešení OpenVPN a strongSwan.

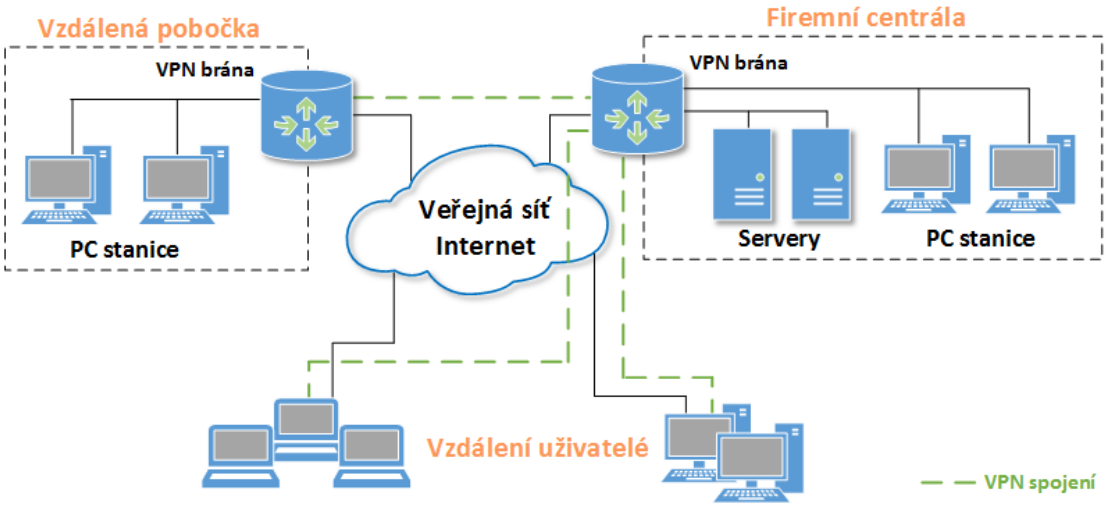
1 Virtuální privátní síť - VPN

V následující kapitole je teoreticky popsána virtuální privátní síť - VPN (Virtual Private Network), důvody a výhody tohoto zavedení a požadavky kladené na VPN. Následně je v kapitole probráno rozdělení VPN, jeho využití a realizace.

Termínem VPN (též nazýváno „tunel" VPN) je myšleno zabezpečené propojení mezi dvěma či více body realizované přes privátní nebo veřejnou síť, v dnešní době nejčastěji Internet. Virtuální síť VPN je tedy v podstatě připojení, které je pomocí šifrovacích a autentizačních technologií zavedeno nad existující již známou infrastrukturou (například sítě LAN – Local Area Network nebo WAN – Wide Area) tak, aby byl zabezpečen užitečný obsah připojení. Tímto se vytvoří virtuální síť mezi dvěma body, které mají k sobě přístup.

Jak již bylo výše zmíněno, ve virtuální privátní síti jde o zabezpečené propojení mezi určitými body, realizované přes síťovou infrastrukturu. Z bližšího pohledu lze toto propojení a funkci mezi dvěma body popsat následovně - na jedné straně je VPN klient, který pomocí protokolů založených na TCP/IP, také označovaných jako protokoly tunelových propojení, virtuálně volá port na druhé straně (server VPN). Při klasickém nasazení virtuální privátní sítě iniciuje klient přes Internet virtuální propojení mezi dvěma body k serveru vzdáleného přístupu. Server VPN vzdáleného přístupu přijme tento požadavek, ověří zadavatele (volajícího) a přenese data mezi klientem VPN na straně jedné a privátní sítí společnosti na straně druhé.

Toto propojení mezi dvěma body se provádí zapouzdřením dat do hlavičky. Tato hlavička obsahuje adresní informace, které umožňují průchod dat přes sdílenou nebo veřejnou infrastrukturu až do koncového bodu. Privátní propojení v síti je prováděno tak, že přenášená data jsou z důvodu utajení šifrována. Pakety zachycené ve sdílené nebo veřejné síti nelze bez šifrovacích klíčů dešifrovat. [1] [2] Podrobnější informace se lze také dočíst například v následující literatuře. [2] Na obrázku 1.1 lze vidět příklad VPN sítě.



Obrázek 1.1: *Příklad VPN sítě*

1.1 Výhody a důvody zavedení VPN

Pro vybudování VPN sítě existuje několik důvodů, které jsou níže popsány včetně výhod a nevýhod tohoto řešení. Společným jmenovatelem těchto důvodů je požadavek „virtualizace" jisté části komunikace v dané organizaci. Jinými slovy, jde o to skrýt jistou část komunikace (případně celou) před veřejnou sítí a jejími uživateli a přitom využít efektivity společné komunikační infrastruktury.

Při budování VPN sítě je dobré nejprve zvážit několik faktorů ovlivňujících toto rozhodnutí jako například jaká je úroveň důvěrnosti (utajení) zasílaných dat nebo jak velký význam je kladen na utajení. V případě kdy je úroveň utajení momentálně dostatečná (zajištěna jiným prostředkem) může být zbytečné používat v takovéto situaci VPN.

Vzdálené spojení v síti lze popsat několika způsoby. Prvním způsobem lze vytvářet nebo si pronajímat vyhrazené vlastní WAN sítě (budovat nákladné privátní sítě). Dalším způsobem je standardní nezašifrovaná komunikace přes Internet a posledním je Internetová komunikace zašifrovaná technologií VPN. V prvním případě při použití vyhrazené, pronajaté (nebo námi vytvořené) linky je výhodou bezpečnost a výkonnost. Toto řešení ale není pro všechny z hlediska financí výhodné, vyhrazená spojení jsou drahá a náklady jsou podstatně větší, než náklady na datové vysokorychlostní připojení při použití VPN. Bohužel při používání sdílené infrastruktury, jako je Internet, je vyvolán větší bezpečnostní problém. Data procházejí celou infrastrukturou sdílenou po celém světě. Výběr VPN je tedy výbornou volbou, neboť využívá funkčnost našeho existujícího připojení k Internetu a zvyšuje bezpečnostní úroveň komunikace. Jedním ze základních důvodů zavedení VPN je tedy ekonomická výhodnost. [2] [3]

1.1.1 Výhody VPN

Hlavní výhodou při používání sítě VPN pro vzdálený přístup je možnost používat veřejně dostupný prostředek (infrastrukturu) pro přenos privátních dat. Jak již bylo výše zmíněno, další výhodou je cenově efektivní řešení. VPN může poskytnout mnoho úrovní zabezpečení, včetně zlepšení důvěrnosti (utajení, integrity a autentičnosti). Jelikož síť VPN využívá již existující infrastrukturu, lze toto řešení rychle realizovat a není nutné čekat na zřízení linky nebo další procedury s tímto vázané. Níže lze vidět stručný přehled několika výhod sítě VPN:

- Bezpečnost - autentizace, autorizace, integrita, různé úrovně šifrování.
- Efektivnost nákladů - šetření financí hned na několika místech.
- Vzdálený přístup - možnost být stále v kontaktu s daty v lokální síti.
- Kontrola - přehled a možnost monitorování přístupů vzdálených klientů.
- Rozšiřitelnost - možnost růstu úměrně potřebám, rozšíření hranic působnosti bez nutnosti budovat další síťovou infrastrukturu.

1.1.2 Nevýhody VPN

U technologie VPN se najdou také jisté nevýhody. Při rozhodování, zda je technologii VPN vhodné zavést do požadovaného prostředí, je nutné tyto nevýhody zvážit. Proces šifrování chrání data a komunikaci, ale také přidává zatížení na síť (např. zatížení přenosové brány) a snižuje využitelnou šířku pásma (zvyšuje se celková šířka pásma potřebná pro šifrovaný přenos). V některých prostředích se mohou vyskytnout problémy s umístěním VPN do existujících pracovišť kvůli další režii s pakety. Pro zvýšení ochrany u VPN mohou být přidány různé úrovně šifrování, nejsilnější šifrování většinou vyžaduje více finančních prostředků (jedná se o náročný proces). Níže lze vidět stručný přehled několika nevýhod sítě VPN:

- Režie spojená s prováděním tunelování - například výměna bezpečnostních informací při vytváření spojení, kontrola spojení.
- Větší náročnost (vyšší režie) spojená s přenosem paketů - přidání dalších potřebných hlaviček k jednotlivým paketům.
- Problémy s realizací (součinnost tunelů VPN a překladu adres NAT).
- Odstraňování těchto nevýhod (problémů) může být někdy složité.
- V některých případech, vyskytující se problém s dostupností internetu. [2]

1.2 Požadavky kladené na VPN

V následující podkapitole jsou popsány požadavky kladené na VPN. Těchto požadavků je hned několik a z různých oblastí, některé vychází již z dříve popsaných důvodů zavedení. Hlavním zaměřením jsou jednotlivé body pro zajištění bezpečnosti.

Jedním z požadavků pro realizaci VPN je rozšiřitelnost, u které jde hlavně o bezproblémový a pohodlný způsob rozšiřování působnosti a přidávání počtu uživatelů. Problém by mohl přijít s nákupem některého prvku, který by měl omezení pro aktuální běh několika tunelů. V případě požadavku většího počtu vytvořených tunelů současně by již bylo potřeba řešit změnu ve VPN síti.

Mezi další požadavky patří také například přívětivost řešení jak uživatelská, tak administrační. Zde je důležitá uživatelská přívětivost (jednoduchost, intuitivnost). V případě složitostí uživatel často hledá vlastní řešení a nedodržuje bezpečnostní pravidla. Dalším požadavkem je schopnost spolupracovat se zavedenými systémy ve společnosti. Například propojení VPN řešení se službou pro správu uživatelů ve společnosti. Dalším z možných požadavků je mobilita - zda lze řešení VPN provozovat na dalších zařízeních (dnešní mobilní telefony), které jsou schopny se připojit do sítě VPN a získat tím neustálý a mobilní přístup.

Další vlastností VPN sítí, na kterou je brán ohled, je přenosová rychlost. Přenosová rychlost je většinou omezena kvalitou a datovou propustností linky, kterou je koncový bod připojen do veřejné infrastruktury. Přenosová rychlost VPN sítě je ale také omezena kvalitou a úrovní použitého šifrování. Na rychlost celého šifrování a tím pádem také VPN síť, má vliv, zda je použit VPN server jako softwarové nebo hardwarové řešení. Náročnost šifrování dále

souvisí s tím, jak náročné šifrovací algoritmy jsou použity. Mezi požadavky při realizaci je také brán ohled na celkovou cenu řešení. Při rozhodování je určitě dobré se dívat nejen na pořizovací náklady, ale i na pozdější investice.

Nyní se již budeme věnovat požadavkům z bezpečnostní oblasti. Součástí sítí VPN je množství bezpečnostních prvků, díky nimž jsou sítě VPN efektivní metodou pro zabezpečení informací. Tyto prvky neboli požadavky, lze přizpůsobit vzhledem k úrovni nedůvěryhodnosti prostředí. Úroveň zabezpečení musí odpovídat důležitosti přenášených dat. Již nižší úroveň šifrování může odradit útočníky (konkurenty) od pokusu získat informace. I méně odolným šifrováním VPN je tedy možné adekvátně ochránit dané informace. Pokud se jedná ale o velmi citlivé a tajné zasílané informace (data), je nutno realizovat silnější způsob šifrování. Bez ohledu na to, jak odolná šifrovací technologie je požívána, by měla tato síť splňovat několik dalších požadavků. Mezi nejdůležitější a nejzásadnější požadavky patří autentizace, autorizace, šifrování a integrita dat. [2]

1.2.1 **Autentizace**

Úkolem autentizace je zajistit, že daný uživatel sítě je skutečně tím, za koho se vydává (slouží tedy k ověření identity uživatele „objektu“ vstupujícího například do nějaké části sítě, systému, do VPN spojení atd.). Autentizace je většinou založena na znalosti autentizačních údajů, jako je jméno, heslo a vlastnictví bezpečnostního (digitálního) certifikátu nebo digitálního podpisu. V této části tedy dochází k ověřování hesla či ověřování validnosti certifikátů. Pokud je celý proces úspěšný, dojde k autentizaci uživatele „objektu“. V dnešní době se stále častěji používá autentizace pomocí dalšího hardware prvku, například pomocí čipových karet a USB Tokenu, kde je uložen bezpečnostní certifikát. Jedná se o tzv. PKI (Public-Key Infrastructure) autentizaci. Pomocí autentizace může síťový administrátor také snadno poznat, kdo se přihlásil k nějaké části sítě, síťovému zařízení či do Internetu, protože každý musí zadat své přihlašovací údaje. [4] [5] [6]

Podrobněji je autentizace probrána v kapitole 2, kde jsou popsány různé metody autentizace, vícefaktorová autentizace se zaměřením na dvoufaktorovou autentizaci ve virtuálních privátních sítích a její výhody. Následně jsou popsány další již zmiňované „prvky“ bezpečnosti. Prvním z nich je autorizace - proces, který následuje po úspěšné autentizaci.

1.2.2 **Autorizace**

Pojmem autorizace se rozumí proces ověření přístupových oprávnění uživatele vstupujícího do informačního systému (v našem případě přístup do vnitřní sítě). Autorizace ve většině případů následuje po úspěšném procesu autentizace. Podstatou autorizace je ověřit, zda daný uživatel má oprávnění provádět určité operace (akce) na softwarovém a hardwarovém prostředku, například vložení nového záznamu do databáze apod. Proces autorizace i autentizace lze provádět vůči lokální databázi uživatelů a také vůči použitému serveru ve společnosti například RADIUS. V návaznosti na zvolenou bezpečnostní strategii bývají oprávnění na provedení těchto akcí rozdělena mezi více subjektů – nejčastěji mezi administrátory, bezpečnostní správce a běžné uživatele. [4] [5]

1.2.3 Šifrování

Šifrování (kryptografie) je jedním z klíčových požadavků na sítě VPN. Jelikož jsou sítě VPN většinou sestaveny nad veřejnou infrastrukturou (Internet), je důležité zajistit dostatečné šifrování. Utajení je zárukou, že nikdo jiný nebude schopen nahlédnout do daných dat. Utajení v sítích VPN zajišťují šifrovací algoritmy, které zpracují data do nesrozumitelných skupin znaků. Pro utajení dat během jejich přenosu veřejnou infrastrukturou jsou data na straně odesílatele zašifrována a na druhé straně u příjemce dešifrována. Šifrování a dešifrování je založeno na tom, že odesílatel i příjemce používají společný šifrovací klíč. Zachycená data odeslaná přes VPN ve veřejné síti jsou nesrozumitelné pro každého, kdo nezná tento společný šifrovací klíč. Délka šifrovacího klíče je důležitým parametrem zabezpečení a pro spolehlivé utajení dat je dobré používat co nejdelší šifrovací klíče [1] [2]. S procesem šifrování souvisí řada pojmů, mezi nejdůležitější patří:

- **Šifrovací algoritmus** je sestaven na matematickém základě a provádí samotné šifrování a dešifrování dat.
- **Šifrovací klíč** říká šifrovacímu algoritmu, jak má data šifrovat a dešifrovat.
- **Šifrovaná data** jsou data, která jsme získali šifrováním.
- **Nešifrovaná data** jsou data, která chceme zašifrovat nebo která jsme získali dešifrováním. [7]

K šifrování se využívají symetrické šifrovací algoritmy, patří zde DES a jeho vylepšená verze 3DES (Triple DES). Dále to jsou například Blowfish, AES a další. Triple DES je oproti novějším algoritmům (například AES) daleko pomalejší, a proto se postupně přestává používat. K výměně šifrovacích klíčů se u PKI autentizace využívá asymetrické šifrování. Nejrozšířenějším příkladem asymetrické šifry je RSA, jako další lze uvést Diffie-Hellman, který je zaměřen na výměnu kryptografických klíčů. Se šifrováním mj. souvisí také digitální podpisy. Do oblasti šifrování patří dále kryptografické kontrolní součty (hash algoritmy), které se aplikují při problému s integritou dat. Pro popis kryptografie, různých šifer a šifrovacích algoritmů doporučuji skripta předmětu bezpečnost v komunikacích [8].

1.2.4 Integrita

Jedním ze základních požadavků je zajištění celistvosti a ochrany dat proti změnám. Právě integrita přenášených dat zaručuje, že data, která jsou obdržena, jsou stejná jako data, která byla vyslána od odesílatele. Tedy že data po cestě nebyla změněna, jak důsledkem poškozeného přenosové trasy, tak například útokem se snahou podvrhnout přenášená data. V dnešní době se k zajištění této integrity dosahuje pomocí hashovací funkce (algoritmu), MACs (Message-authentication codes) a digitálních podpisů. [2]

- **Hashovací funkce** se také nazývají jednosměrné funkce, protože lze snadno určit hodnotu hash ze zprávy, ale je nemožné matematicky určit z hashované zprávy původní zprávu. Hashovací funkce s využitím klíče generuje přesně danou velikost souboru, která je založena na jeho skutečné velikosti. Ověření na straně příjemce je provedeno výpočtem nového hodnoty (hashe) a jejího porovnání s poslanou hodnotou od odesílatele. Pokud se zpráva během přenosu změnila, hodnoty algoritmu hash se budou lišit a data budou odmítnuta. Hashování dvou různých zpráv vytvoří dvě různé hashované zprávy. Příklady hashovacích algoritmů jsou MD5 (Message Digest 5), SHA (Secure Hash Algorithm), kde SHA je považováno za bezpečnější než MD5, protože vytváří 160 bitové hashované zprávy oproti 128 bitovým u MD5.
- **MAC (Message-authentication code)** - v některých literaturách se můžeme také setkat s označením MIC (Message Integrity Code). My se zaměříme na typ kódu HMAC (Hashed Message Authentication Code), což je typ autentizačního kódu zprávy (MAC) počítané s použitím hashovací funkce v kombinaci s tajným šifrovacím klíčem. Kódy HMAC tedy zajišťují integritu pomocí algoritmu hash s klíčem, což je výsledek matematického výpočtu u zprávy, který využívá funkci hash v kombinaci se sdíleným tajným klíčem. Síla kódu HMAC přímo závisí na síle hashovací funkce, velikosti a kvalitě klíče. Velikost výstupu HMAC kódu je stejná jako velikost výstupu hashovací funkce. HMAC je často spojován s hashovacími funkcemi SHA1 nebo MD5.
- **Digitální podpis -** využívá kombinaci hashovacího algoritmu a asymetrické šifry. U digitálního podpisu odesílatel podepíše dokument s jeho privátním klíčem a příjemce poté ověří tento soubor pomocí odesílatelova veřejného klíče. [9] [10] Více o hashovacích funkcích (algoritmech) a digitálních podpisech se lze dočíst například v následující knize věnované bezpečnosti [2] a skriptech pro bezpečnost v komunikacích [8].

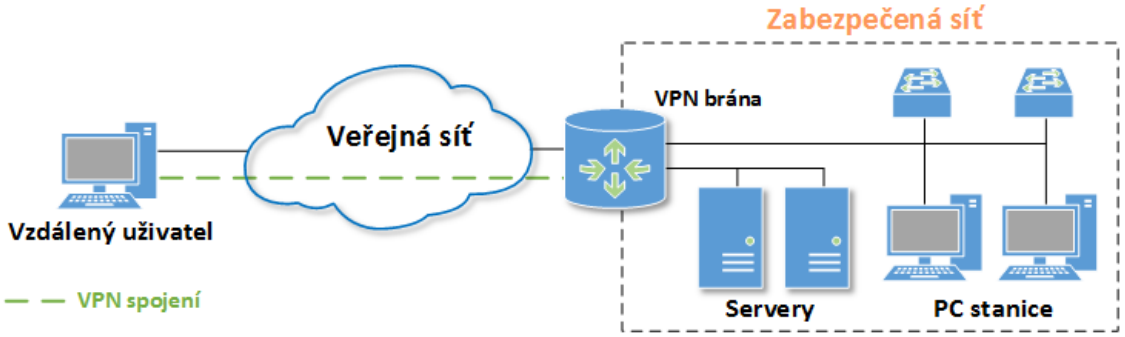
1.3 Rozdělení VPN dle typu spojení

VPN je velice široký pojem a zahrnuje řadu protokolů, technologií a v různých odborných literaturách se lze setkat s rozdílným označením jednotlivých typů. Jak již bylo řečeno, VPN síť lze vytvořit v prostředí veřejné sítě (sdílené infrastruktury), nejčastěji Internet. Pro vytváření VPN sítí máme tedy prostředí Internetu, kdy VPN síť je vytvořena právě přes tuto veřejnou síť. Mezi další prostředí lze také zařadit sítě Intranet a Extranet, které jsou popsány níže. Následná část je zaměřena na popis rozdělení VPN spojení a to na dva základní typy - připojení pro vzdálený přístup a připojení mezi sítěmi.

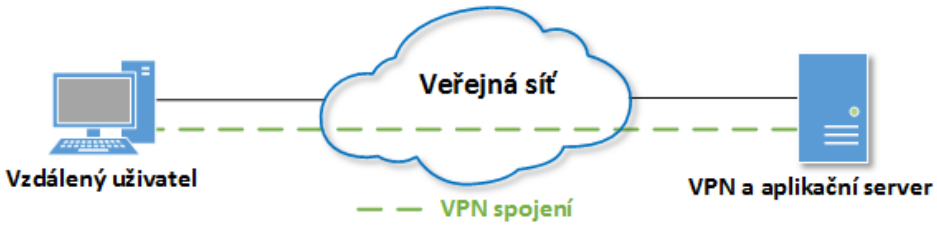
1.3.1 Připojení VPN pro vzdálený přístup

Tento typ sítě VPN (anglicky Remote-Access) umožňuje bezpečné připojení jednotlivých uživatelů pracujících z domova nebo mobilních uživatelů do privátní sítě pomocí veřejné infrastruktury. Pomocí tohoto vzdáleného přístupu mohou uživatelé využívat většinu

firemních služeb. Vzdálené stanice takto připojené do sítě VPN se chovají jako součást privátní sítě. Na počítači či jiném uživatelském zařízení se využívá softwarový VPN klient, díky kterému je zřízen zabezpečený tunel. Přesná fyzická infrastruktura sdílené sítě není podstatná, z logického pohledu spojení to vypadá, že jsou data odesílána přes vyhrazenou privátní linku. VPN pro vzdálený přístup lze dále rozdělit dle logického členění na: VPN typu zařízení - síť a zařízení - zařízení. První zmiňované jsme si pospali již výše, kde se připojíme do požadované privátní sítě. U druhého typu se vzdáleně připojíme jen ke konkrétnímu zařízení, například spojení klient - server, kdy se využívá bezpečnosti VPN k realizaci citlivých operací na serveru. [1] [11] Na obrázku 1.2 a 1.3 se nachází jednotlivé typy vzdáleného připojení a rozdíl mezi nimi.



Obrázek 1.2: *Spojení zařízení - síť*

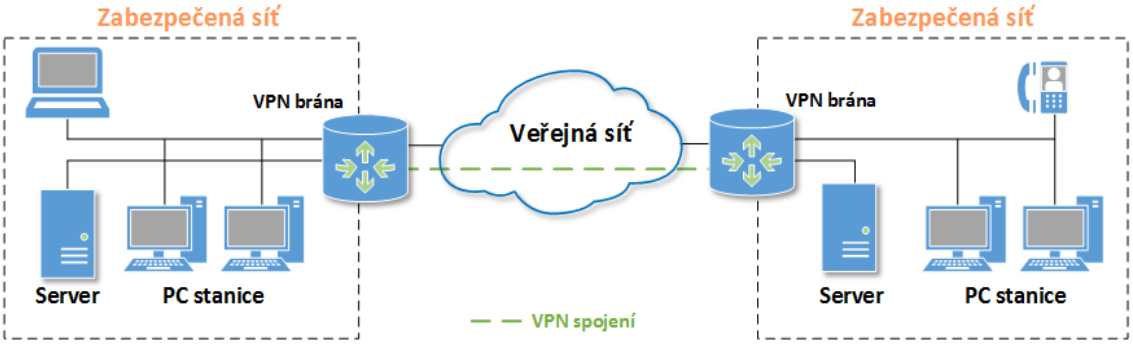


Obrázek 1.3: *Spojení zařízení - zařízení (klient - server)*

1.3.2 Připojení VPN mezi sítěmi

Připojení VPN mezi sítěmi (anglicky označováno site-to-site) umožňuje organizacím propojení mezi odloučenými pobočkami nebo jinými organizacemi bez ohledu na jejich geografickou pozici přes veřejnou síť při zachování zabezpečené komunikace. Připojení VPN mezi sítěmi je ideální volbou všude, kde je potřeba jednotné spolupráce mezi zaměstnanci. Tento typ VPN tedy propojuje celé sítě odloučených poboček. Při tomto VPN spojení jsou využívány hardwarové prostředky, je většinou realizováno pomocí firewallů nebo směrovačů.

Při propojení sítí poboček přes Internet, jak znázorňuje následující obrázek 1.4, předává směrovač pakety dalšímu směrovači přes připojení VPN, toto propojení přes Internet funguje logicky jako vyhrazená linka.



Obrázek 1.4: *Připojení VPN mezi sítěmi*

Připojení VPN mezi sítěmi můžeme dále rozdělit dle prostředí pro vytváření VPN sítí na Intranet a Extranet, které jsou také označovány jako site-to-site Intranet a site-to-site Extranet. Níže je již popsán rozdíl mezi těmito dvěma typy.

- **Připojení VPN mezi sítěmi – Intranet.** VPN síť je vytvořena v rámci firemní sítě, připojení je tedy využíváno pro propojení geograficky odlehlých poboček společnosti.
- **Připojení VPN mezi sítěmi – Extranet.** Zde je VPN síť vytvořena mezi sítěmi různých firem a organizací. Poskytuje bezpečné spojení s obchodními partnery, dodavateli a zákazníky, například za účelem vedení elektronické komerce.

Při porovnání připojení VPN pro vzdálený přístup a mezi sítěmi je základním rozdílem to, že u vzdáleného přístupu je nutná aktivita od uživatele, který musí vytvořit spojení. U připojení mezí sítěmi je spojení realizováno například na směrovači. Pro uživatele na pracovišti je VPN spojení s odloučeným pracovištěm neustálé, uživatel nemusí spojení navazovat. Dalším rozdílem jsou používané typy protokolů. Při použití vzdáleného přístupu se dnes využívá hlavně protokol SSL/TLS (obecně SSL VPN). Pro připojení mezi sítěmi se především využívá protokol IPsec, který je v dnešní době bezpečnostním standardem.

1.4 Rozdělení VPN dle použitého protokolu

V následující kapitole je krátce popsáno rozdělení VPN dle použitých protokolů. Zaměřuji se zde na popis protokolu IPsec a SSL/TLS (SSL VPN), neboť právě použitá řešení v praktické části využívají těchto protokolů. Komunikace a data mohou být zabezpečena šifrováním na mnoha odlišných vrstvách síťového modelu ISO/OSI, bývají to vrstvy aplikační, transportní, síťová a linková.

Na aplikační vrstvě může být šifrování zajištěno pomocí zabezpečených kanálů, jako je SSH. Co se týče transportní vrstvy, zde může být použit například protokol SSL pro ochranu užitečného obsahu komunikace mezi uživateli. Na síťové vrstvě je to protokol GRE vyvinutý původně firmou Cisco, který je určený k zapouzdření paketů jednoho protokolu do protokolu jiného (tunelování). Dále je to již zmíněný IPsec protokol. Na druhé linkové vrstvě je to tunelovací protokol L2TP (Layer 2 Tunneling Protocol), který je nadstavbou tunelovacího protokolu mezi dvěma body (Point-to-Point Protocol - PPP). Protokol L2TP je nástupcem

PPTP (Point-to-Point Tunneling Protocol). Protokol L2TP se často pro větší bezpečnost kombinuje s IPsec.

V případě získání více informací o výše zmíněných protokolech a probrání detailní funkčnosti využít například literaturu Bezpečnost v počítačových sítích [2] a příslušné RFC dokumentace jednotlivých protokolů.

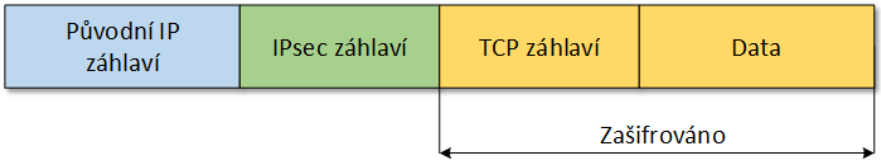
1.4.1 IPsec protokol

IPsec (Internet Protocol Security) je standardizovaná skupina protokolů pro zajištění autentizace, integrity dat a šifrování IP komunikace mezi dvěma koncovými systémy, technicky realizovanou pomocí dynamicky navazovaných zabezpečených tunelů na síťové vrstvě referenčního modelu ISO/OSI, doplňuje IPv4 protokol (v IPv6 je povinnou součástí protokolu). IPsec je jedna z nejrozšířenějších VPN technologií a otevřený standard (RFC 4301). IPsec jako první zařídí to, že se obě koncové strany navzájem identifikují (autentizují) a následně se šifruje komunikace pomocí domluveného algoritmu. IPsec není závislá na konkrétních algoritmech, neurčuje, jaké algoritmy se musí používat pro komunikaci. Definuje ale mechanismy vyjednávání a základní množinu algoritmů.

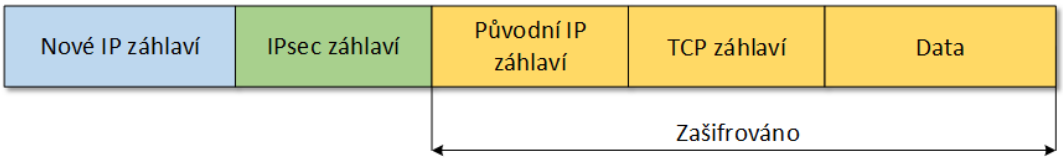
Skupina parametrů dohodnutých pro šifrování a autentizaci mezi konci tunelu se nazývá Security Association (SA), v překladu bezpečnostní asociace. SA je v podstatě skupina algoritmů, které poskytují parametry pro bezpečnou komunikaci pomocí AH a ESP, obsahuje informaci o šifrovacích a autentizačních algoritmech a příslušné klíče. Pro každý směr provozu v tunelu je dohodnuto zvláštní SA s omezenou platností. Dohoda SA mezi konci tunelu se realizuje pomocí protokolu IKE (Internet Key Exchange), což je konkrétní implementace protokolu pro bezpečnou dohodu klíčů vyhovující obecnému rámci pro protokoly tohoto určení nazývanému ISAKMP (Internet Security Association and Key Management Protocol). Parametry pro autentizaci a šifrování jednotlivých paketů se přenášejí přímo v těchto paketech v pomocných hlavičkách, nazývaných Authentication Header (AH) a Encapsulating Security Payload (ESP). AH zajišťuje integritu a autentizaci zdroje dat, využívá hashovací funkce jako MD5 nebo SHA1 a společný klíč. Autentizace se zde vztahuje i na část vnějšího IP záhlaví (na rozdíl od ESP), ale nepoužívá se zde šifrování dat. ESP je novější a používanější hlavička, může zajistit všechny funkce jako AH a navíc ještě šifrování. Využívá šifrovací algoritmy jako DES nebo AES. [12]

Proces vyjednávání IPsec tunelu probíhá ve dvou fázích. V první fázi se pomocí IKE protokolu autentizují účastníci a vyjedná se IKE SA pomocí Diffie-Hellmanova protokolu, který je použit pro výměnu klíčů. Díky fázi jedna se vytvoří zabezpečený kanál pro vyjednání IPsec SA ve fázi dva. Fáze dva vyjednává IPsec SA parametry a nastaví odpovídající bezpečnou asociaci SA. Technologie IPsec umí ale zabezpečit jen unicastový provoz, pro multicastový a broadcastový provoz se použije jako první zapouzdření pomocí GRE protokolu a až poté pomocí IPsec. Výše jsou popsány základní vlastnosti a funkce protokolu IPsec, ovšem pro detailní popis průběhu protokolu IPsec a všech pojmů doporučuji literaturu [2], kde je vše rozebráno do hloubky včetně příkladu.

U VPN sítí je možné se setkat se dvěma základními módy činnosti. Taktéž IPsec může pracovat v jednom z těchto dvou módů, jedná se o transportní a tunelovací mód. Transportní mód šifruje pouze data, původní záhlaví paketu se nemění. IP hlavička se tedy ponechá a doplní se pouze IPsec hlavička. Tunelovací mód šifruje celý paket (včetně hlavičky), paket je zapouzdřen a doplněn novým záhlavím (hlavičkou). Na obrázcích 1.5 a 1.6 můžeme vidět rozdíl mezi oběma módy. [11]



Obrázek 1.5: *Transportní mód*



Obrázek 1.6: *Tunelovací mód*

1.4.2 SSL protokol

Na vyšších vrstvách než IPsec působí další bezpečnostní protokol a to SSL (Secure Socket Layer) a jeho následovník TLS (Transport Layer Security), které zajišťují šifrování přenášených dat. Ve verzi TLS 1.0 jde v podstatě o SSL 3.0, kde jsou drobné rozdíly a vylepšení, ale v zásadě jsou stejné. Zkratka SSL je běžně používaná k označení protokolů SSL i TLS, někdy je možné se setkat také se zápisem SSL/TLS. SSL protokol zajišťuje autentizaci komunikujících stran, integritu a šifrování aplikačních dat při jejich přenosu přes veřejnou IP síť. SSL vyžaduje spolehlivý transportní protokol (TCP).

Z pohledu TCP/IP protokolu běží SSL na aplikační vrstvě. Ovšem SSL není protokolem jedné vrstvy, z pohledu modelu ISO/OSI se SSL nachází mezi aplikační a transportní vrstvou, která je někdy také chápána jako součást prezentační vrstvy. SSL nezabezpečuje veškerou komunikaci jako v případě IPsec, ale pouze některé aplikace. SSL dává klientskému počítači možnost provozovat zabezpečené služby pouze pro určité aplikace. Také se využívá pro vzdálený přístup, včetně virtuálních privátních sítí (SSL VPN), kde je základem pro zajištění bezpečnosti právě protokol SSL/TLS. To že není zabezpečena veškerá komunikace, může být pro budování VPN sítí slabší faktor z hlediska bezpečnosti. Následně je popsán obecný princip SSL. V protokolu SSL jsou využívány asymetrické i symetrické šifrovací algoritmy. Před vlastním přenosem dat si obě komunikující strany, vstupující do SSL spojení, ověří totožnost. Při této fázi autentizace SSL využívá pro výměnu klíčů asymetrické kryptografie (kombinace šifrování veřejným a soukromým klíčem). Po této fázi autentizace si strany vymění informace, díky kterým je sestaveno SSL spojení. Vlastní přenos SSL dat je šifrován symetrickými algoritmy. Integrita dat je zde zajištěna hashovacími funkcemi, například SHA nebo MD5. SSL podporuje obousměrnou autentizaci, ale používá se většinou pouze jednosměrně. Lze využít

jména a hesla, jména a token nebo digitální certifikáty X. 509. Klient SSL spoléhá na digitální certifikát od vzdáleného serveru, k němuž se chce připojit. V dnešní době SSL VPN začínají nahrazovat VPN pro vzdálený přístup založené na IPsec. Výhodou VPN založené na tomto protokolu je, že SSL VPN využívá pro svůj provoz standardně povolený TCP port 443, není tedy problém s připojením z jakékoli části sítě. SSL VPN dnes již neslouží pouze k přístupu do firemní sítě skrze webový prohlížeč, ale hojně se využívá a umožňuje plnohodnotný přístup do firemní sítě pomocí VPN klienta. Dnes asi nejrozšířenější variantou pro spojení VPN mezi sítěmi, využívající SSL (OpenSSL), je open source řešení OpenVPN. [13] [14] [15] Nyní jsme se seznámili se základními vlastnostmi SSL, pro více informací doporučuji RFC dokument (Transport layer Security, RFC 2246) nebo také literaturu Bezpečnost v komunikacích [8].

1.5 Možnosti realizace

Možnosti realizace (řešení) VPN sítí lze rozdělit na dva základní okruhy dle zvolné platformy a to na hardwarové a softwarové VPN. Obecně lze říci a je možné se dočíst, že hardwarové řešení bývá výkonnější jak z pohledu propustnosti, tak z pohledu šifrování. Hardwarové řešení je finančně nákladnější, ale má výkonnější řešení. Některé dražší specializované zařízení bývají osazovány speciálními čipy pro akceleraci šifrování. Při koupi hardwarového řešení často zakoupíme „box“, který je ve funkci směrovače, firewallu a VPN koncentrátoru. Nejčastěji se dá setkat se zařízeními například od společnosti Cisco nebo Juniper.

Softwarové řešení je levnější a v mnoha případech pro menší podniky dostupnější variantou. V dnešní době jsou nabízeny jak placené řešení, tak spousta řešení zdarma (open source software). Mezi nejrozšířenější řešení zdarma lze zařadit OpenVPN a strongSwan. Řešení OpenVPN a strongSwan je díky podpoře a otevřenosti kódu často nasazováno a používáno na platformách Linux. Obě tato řešení jsou podrobněji popsána později a taktéž je na těchto dvou řešeních stavěna praktická část, kde se podrobně věnuji jejich konfiguraci.

2 Vícefaktorová autentizace

V podkapitole 1.2.1 byla krátce popsána autentizace. Jen pro připomenutí, už tedy víme, že hlavním úkolem autentizace je ověření identity uživatele „objektu“ vstupujícího například do nějaké části sítě, systému nebo VPN spojení. Autentizace je většinou založena na znalosti autentizačních údajů, nejčastěji jako je jméno a heslo, vlastnictví bezpečnostního certifikátu nebo digitálního podpisu. Představme si ale situaci, kdy chceme zašifrovat nějaká data na našem počítači. Použijeme k tomu drahý šifrovací software, který využívá nejlepší šifrovací algoritmy. Data jsou bezpečně zašifrována a lze se k nim dostat jen pomocí šifrovacího klíče, který je bohužel uložen také na počítači. Aby to bylo bezpečné, uživatel se k šifrovacímu softwaru přihlašuje pomocí jména a hesla (šifrovací klíč je tedy chráněn ještě heslem). A zde narážíme na největší problém, celá bezpečnost je závislá na našem zvoleném hesle, nikoliv na téměř neprolomitelném softwaru nebo jiných řešeních. Proto je dobré ukládat šifrovací klíče (také certifikáty) mimo harddisk na nějaké bezpečné zařízení, kam se útočník dostane jen velmi těžce.

Je důležité si také uvědomit, že se většina hesel (k emailu, informačnímu systému atd.) dá odchytit už na úrovni klávesnice. Do paměti se umístí program, který bude ukládat do souboru všechny stisknuté znaky. Je tedy velmi dobrý důvod k autentizaci začlenit další prvek (faktor) neboli metody autentizace, které jsou probrány v další podkapitole. [4] [20]

2.1 Metody autentizace

Následující podkapitola je zaměřena na rozdělení metod autentizace. Nezávislé autentizační faktory (metody) lze rozdělit do třech kategorií, které se liší způsobem, jakým ověřují totožnost uživatele. Jednotlivé varianty je možné pak kombinovat s celou řadou technologií lišících se uživatelským pohodlím a spolehlivostí, s jakou dokážou přesně určit, koho do systému pustit, a koho ne. Mezi metody přihlášení pak patří nejčastěji následující kombinace:

- **Znalost** - něco vím, něco co uživatel musí znát. Typická jsou přístupová hesla, správná kombinace znaků, pro bankomaty nebo mobilní telefony PIN kódy a také správné odpovědi na „bezpečnostní otázky“ nebo šifrovací klíč.
- **Vlastnictví** - něco mám, autentizovat se může ten, kdo vlastní nějaký předmět. Patří mezi ně fyzické klíče, průkazy totožnosti a také komunikační zařízení, například hardwarový token (USB token), bezpečnostní čipové karty, standardní mobilní telefon nebo smartphone.
- **Biometrika** - něco jsem, tato třída zahrnuje využívání biometrických senzorů pro snímání otisků prstů, sítnice a duhovky nebo algoritmy pro měření charakteristiky chování, jako rytmus psaní nebo identifikace hlasu.

Jak již bylo řečeno v úvodu kapitoly, často používaná a běžná autentizace je založena na znalosti uživatelského jména a hesla. Heslo by si měl uživatel pamatovat, ale právě z tohoto důvodu uživatelé často volí snadno zapamatovatelná, tzn. „slabá“ hesla. Jako ověřovací faktor zde slouží jediný faktor - heslo. Aby se předešlo a zamezilo úniku a krádeži dat, používá se vícefaktorová autentizace. Jak již z předešlé strany vyplývá, tímto termínem rozumíme současné užití alespoň dvou různých faktorů (dvoufaktorová autentizace). Třífaktorová autentizace je poté využití například tokenu, znalosti i biometriky zároveň.

Níže na tabulce 2.1 je možné vidět základní druhy autentizace, kdy se využívá kombinace právě zmíněných tří metod autentizace (znalost, vlastnictví a biometrie). Dostáváme tedy celkově sedm možností. První možnost založená pouze na znalosti není ideální řešení. K druhé možnosti (pouze vlastnictví) patří různé bezkontaktní tokeny, karty, které nejsou chráněny žádným heslem. Při krádeži tedy hrozí zneužití a také duplikace předmětu.

Tabulka 2.1: *Základní možnosti autentizace*

	1	**2**	**3**	**4**	**5**	**6**	**7**
Znalost	•			•		•	•
Vlastnictví		•		•	•		•
Biometrika			•		•	•	•

Možnost tři je první, dá se říct rozumnou, (bezpečnou) variantou k využití pro autentizaci ve světě počítačů. Ovšem levnější snímače otisků prstů nijak zvlášť bezpečné nejsou a lze je obejít. Šestá možnost je kombinace právě biometrie a ještě navíc použití hesla. Zbývají nám už jen možnosti 4, 5 a 7. Právě tyto možnosti obsahují token, který je zabezpečen proti zneužití při ztrátě. U možnosti 4 máme typicky klasickou čipovou kartu, USB token nebo autentizační kalkulátor. Uživatel se zde prokazuje vlastnictvím tokenu a přístup k tokenu je navíc chráněn heslem. Základní idea autentizačního kalkulátoru je ta, že uživatel zadá heslo a po zadání kalkulátor vygeneruje jednorázové přístupové (pokaždé jiné) heslo. Možnosti 5 a 7 jsou případy, kdy se autentizujeme vůči tokenu biometrickou veličinou, nejčastěji otiskem prstu, nebo otiskem prstu a heslem. Zvláště možnost 7 při použití všech tří faktorů je více finančně náročnější než dvoufaktorová autentizace pomocí tokenu a hesla. Většina čipových karet a USB tokenů jsou dnes chráněna heslem. [20]

2.2 Dvoufaktorová autentizace

Dvoufaktorová autentizace je autentizace, kdy se ke standardnímu heslu přidá jeden faktor navíc, který sníží riziko bezpečnostního incidentu. Nejčastějším a neznámějším příkladem využití dvoufaktorové autentizace je kombinace faktoru „něco vím“ ve formě hesla se zasíláním SMS zpráv na mobilní telefon (něco mám) nebo využitím aplikace pro tvorbu jednorázových hesel ve smartphonech. Hlavní výhoda tohoto systému spočívá v tom, že mobilní telefon vlastní skoro každý, a odpadá tak nutnost koupit si nebo instalovat novou platformu, která by sama o sobě plnila pouze funkci dalšího autentizačního faktoru. Ke svému profilu se

tedy přihlásíme pouze v případě, že známe heslo a máme u sebe svůj mobilní telefon. [6] My se ovšem budeme v této práci převážně zabývat dvoufaktorovou autentizací s využitím USB tokenu a bezpečnostních certifikátů. Co se týče obecných požadavků kladených na USB tokeny, jedná se například o odolnost tokenu vůči vnějším vlivům (vodotěsnost) a odolnost konektorů. Výhodou USB tokenu oproti použití čipových karet je mobilita (nepotřebujeme pokaždé speciální čtečku). Dalším parametrem bývá paměťová kapacita tokenu. Mezi bezpečnostní požadavky lze zařadit, zda je token chráněn heslem nebo zda nabízí různé typy zablokování při opakovaně chybném zadání hesla. Dále lze například uvést podporu šifrování v hardwaru tokenu, či jak je řešen import/export klíčů a certifikátů. Token může mít hardwarovou podporu různých šifrovacích algoritmů a hash funkcí.

2.2.1 Některé výhody systému dvoufaktorového zabezpečení

- Zabránění jednofaktorovým útokům.
- Uživatelé jsou zodpovědní za všechny akce, které vychází z úspěšné autentizace.
- Mnohem nižší pravděpodobnost, která vede k podvodnému přístupu k datům.
- Jednoduché pro použití koncovými uživateli.
- Trvanlivé a nabízené řešení dlouhodobé bezpečnosti.
- Jednoduché na správu.

2.2.2 VPN a dvoufaktorová autentizace

Pokud využívá firma či organizace VPN síť, určitě je vhodné zvážit dvoufaktorovou autentizaci. Celá VPN je totiž bezpečná pouze jako její nejslabší článek a tím je způsob autentizace a jeho náchylnost na odcizení identity uživatelů. Pro dvoufaktorovou autentizaci je kromě serveru a klientského prostředí, které je nainstalováno na všech přístrojích se kterými se chceme připojovat k podnikovým zdrojům, zapotřebí autentizační předmět. Jako autentizační předměty slouží již například zmiňované USB tokeny, čipové karty a softwarové aplikace, které má uživatel na svém notebooku, tabletu či v mobilním telefonu.

Dnes se můžeme setkat s různými nabídkami a řešeními dvoufaktorové či vícefaktorové autentizace. Jsou to řešení nabízená různými společnostmi, které vytvářejí pro firmu vše na zakázku, nabízejí servis, zákaznickou podporu a další. V takovémto případě je nabízené řešení například nainstalování speciální autentizační brány do podniku. S tím souvisí i používání speciálního USB Tokenu nebo čipové karty, která je ověřována vůči autentizační bráně. Ovšem tato řešení nepatří mezi nejlevnější. Dalším řešením pro zabezpečení je využití open source řešení například OpenVPN či strongSwan. Obě řešení podporují dvoufaktorovou autentizaci, a to jak pomocí USB tokenů, tak čipových karet. Stačí tedy tyto open source řešení nainstalovat na server v podniku, nakonfigurovat dle požadavků a zakoupit například USB tokeny pro uživatele. Více informací k OpenVPN a strongSwan se nachází v dalších kapitolách praktické části a také na stránkách těchto řešení [16] [19].

3 Použitý open source software

V následující kapitole jsou teoreticky probrána open source řešení OpenVPN a strongSwan. Popisuji zde základní vlastnosti a možnosti těchto řešení. OpenVPN a také strongSwan jsou použity pro praktickou realizaci této práce, tedy vytvoření VPN sítě a zavedení dvoufaktorové autentizace.

3.1 OpenVPN

OpenVPN je volně dostupný (open source) software včetně zdrojového kódu vytvořený v roce 2002 Jamesem Yonanem a publikován pod licencí GNU General Public License (GPL). Používá se k vytváření VPN sítí mezi různými lokalitami, dokáže tedy vytvořit šifrovaný VPN tunel mezi požadovanými stanicemi (lokalitami). Spadá do skupiny open source VPN, které nepoužívají protokol IPsec, místo toho je použit protokol SSL/TLS. Co se bezpečnosti týče, umí toho OpenVPN opravdu hodně. Podporovány jsou režimy se sdíleným klíčem, použití SSL/TLS certifikátů nebo uživatelské jméno a heslo. Samozřejmostí je možnost použít libovolné šifrovací algoritmy, které podporuje použitá OpenSSL knihovna.

OpenVPN je založeno na architektuře klient - server, musí být tedy nainstalován na obou koncích VPN, kde jeden je určen jako server a druhý jako klient. Co se týče podpory operačních systémů, je opravdu široká, OpenVPN může být nainstalováno na systémech Linux, Windows, OpenBSD, FreeBSD, Mac OS X a Solaris. Linuxové systémy musejí mít kernel verze 2.4 a vyšší. Je podporováno také mnoho mobilních systému jako Android, iOS nebo Windows Phone. OpenVPN nabízí také placené VPN řešení OpenVPN Access Server, které obsahuje webovou správu pro administraci. Další nabízenou službou postavenou na OpenVPN je takzvaný PrivateTunnel, kdy při registraci dostáváme zabezpečený přístup skrze Internet. Využití přenosu dat do 100MB je zdarma, při větším přenosu dat si musíme zakoupit některý z nabízených tarifů.

OpenVPN standardně používá protokol UDP, ale lze použít i TCP. Veškerá komunikace probíhá na jediném portu, a je tedy možné snadno nakonfigurovat firewall tak, aby propouštěl pouze pakety na tomto portu. Stejně tak je možné, aby pakety procházely NATem. Na základě oficiálního přidělování portů organizací IANA je předvolený port stanoven jako UDP port 1194. Nicméně můžeme použít jakýkoli jiný TCP nebo UDP port, od verze 2.0 pak na serveru OpenVPN může být jediný port použit pro několik tunelů.

Celý OpenVPN proces (démon) běží v uživatelském režimu a komunikuje s operačním systémem skrze virtuální síťová rozhraní TUN nebo TAP. Při vytváření VPN sítě si tedy můžeme vybrat mezi implementací přes TAP rozhraní (Ethernet - přepínání) nebo přes TUN rozhraní (IP - směrování). OpenVPN je dobrou alternativou k protokolu IPsec, který používá své vlastní protokoly. Tohoto faktu zneužívají někteří ISP, kteří je blokují a snaží se tak uživatelům vnutit dražší služby. Důležité je také zmínit, že zde chybí kompatibilita s IPsec nebo jinými implementacemi VPN. [15] [16] [17]

3.2 strongSwan

StrongSwan je volně dostupný (open source) a jeho zakladatelem je Andreas Steffen, který je profesorem na bezpečnost v komunikaci a vedoucí Institutu pro internetové technologie a aplikace na Univerzitě aplikovaných věd Rapperswil. Toto řešení se používá k vytváření VPN sítí (šifrovaných VPN tunelů) mezi různými lokalitami. Je popisováno jako potomek projektu FreeS/WAN, strongSwan je nadále uvolněn a publikován pod licencí GNU General Public License (GPL). Jedná se open source řešení, které využívá a implementuje protokol IPsec na rozdíl od OpenVPN, kde se využívalo SSL/TLS. Podpora operačních systémů je o něco menší než v případě řešení OpenVPN, strongSwan je dostupný pro systémy Linux (2.6 a 3.x jádra), FreeBSD a Mac OS X. Podpora pro operační systém Windows donedávna chyběla, strongSwan od verze 5.2.0 podporuje sestavení na OS Windows pomocí MinGW toolchain, ale využití je atím značně omezeno více o této podpoře v kapitole 7.1. Dále jsou samozřejmě podporovány mobilní systémy jako například Android nebo Maemo.

U řešení strongSwan se snaží klást důraz na silné šifrování a autentizaci. Co se bezpečnosti týče, jsou zde využívány autentizační metody pomocí sdíleného klíče nebo X. 509 certifikátů, možnost volitelného bezpečného uložení soukromých klíčů na čipové karty a tokeny pomocí standardizovaného PKCS # 11 rozhraní. Dále také podpora autentizace pomocí uživatelského jména a hesla. Je kladen důraz na silnou politiku IPsec podporující velké a složité VPN sítě. StrongSwan nabízí podporu pluginů pro zvýšení jeho funkčnosti. Samozřejmostí je možnost použít různé šifrovací algoritmy, které jsou dostupné pro toto řešení. Uživatel si může vybrat mezi třemi kryptografickými knihovnami legacy FreeS/WAN, OpenSSL a gcrypt. Implementuje obě verze protokolu IKEv1 a IKEv2. StrongSwan verze 5.0 plně implementuje IKEv2 protokol. Hlavní změnou, která si vysloužila nové číslování (verze 5.0), je odstranění IKEv1 démona pluto. Tímto krokem se strongSwan zbavil kódu odvozeného z původního projektu FreeS/WAN, ze kterého rovněž vychází Openswan. Tomuto kroku předcházela implementace IKEv1 pro démona charon, který byl už v předchozích verzích používán pro protokol IKEv2. Pro vývojáře to znamená možnost soustředit veškeré síly na vývoj jediného víceprotokolového démona. Je zde plná podpora pro vytváření IPsec IPv6 tunelů. Důležité je také zmínit, že strongSwan spolupracuje s jinými implementacemi IPsec, včetně různých klientů. [18] [19]

4 Generování certifikátů

Každou chvíli si lidé pořizují digitální certifikát u některé certifikační autority. U nás v České Republice se jedná o následující certifikační autority neboli kvalifikované poskytovatele certifikačních služeb - První certifikační autorita (I. CA), PostSignum nebo eIdentity. Rolí certifikační autority (zkratka CA) v rámci elektronické komunikace je být třetí nezávislou stranou. CA je v asymetrické kryptografii subjekt, který vydává digitální certifikáty. Tímto usnadňuje využívání PKI (Public Key Infrastructure) tak, že svojí autoritou potvrzuje pravdivost údajů, které jsou ve volně dostupném veřejném klíči uvedeny. CA však umožnují a vykonávají i další úkony, například autentizaci a registraci ostatních certifikačních autorit a uživatelů, vydávání seznamu zneplatněných certifikátů a další. Služby certifikačních autorit jsou zpoplatněné. Pro vygenerování žádosti o certifikát používá drtivá většina žadatelů generátor klíčů poskytovaný příslušnou autoritou. Na internetu lze ale najít i další alternativní generátory klíčů, které zastanou stejnou práci a umožnují nám také vytvořit svou vlastní CA. Jedním z nich je například program XCA (nebo Easy RSA). V této práci bude využívána právě vlastní vytvořená certifikační autorita, která je vhodná pro testování a pomoci které vydáme a podepíšeme potřebné certifikáty. Pro podrobnější informace a více o problematice certifikátů a certifikační autoritě doporučuji například literaturu [25].

Dále je odkazováno na přílohu A, která popisuje generování certifikátu a vše s tímto spojené, pomocí software Easy RSA a XCA (X Certificate and key management). V této příloze jsou popsány základní vlastnosti a možnosti těchto softwarů, včetně postupu a vytvoření požadovaných certifikátů pro pozdější použití v řešeních OpenVPN a strongSwan.

V tabulce 4.1 je přehled vygenerovaných souborů a jejich charakteristika. Tabulka ke každému souboru znázorňuje pro koho je určen, jaký je jeho účel a zda musí být udržen v tajnosti. K vygenerování těchto souborů dojdeme pomocí nástroje Easy RSA nebo softwaru XCA.

Tabulka 4.1: *Vlastnosti vygenerovaných souborů [26]*

Název souboru	**Potřebný pro**	**Účel**	**Tajný**
ca.crt	server + všechny klienty	Certifikát kořenové CA	Ne
ca.key	pouze CA	Klíč kořenové CA	Ano
dh{n}.pem	pouze server	Parametry D-H algoritmu	Ne
server.crt	pouze server	Certifikát serveru	Ne
server.key	pouze server	Klíč serveru	Ano
klient.crt	pouze klient	Certifikát klienta	Ne
klient.key	pouze klient	Klíč klienta	Ano

5 Instalace USB tokenu

Již několikrát byl v této práci zmíněn pojem USB token. USB token je hardwarové zařízení, sloužící k uložení digitálních certifikátů, veřejných a privátních klíčů. Při využívání v praxi se tento token připojí pomocí USB konektoru k počítači a chráněná data uložená na USB tokenu slouží k autentizaci konkrétního uživatele. USB token bývá ještě zabezpečen pomocí PIN a PUK kódu, který při ztrátě zařízení chrání uložená data. Na obrázku 5.1 lze vidět USB token Aladdin eToken PRO 32k, který bude použit při praktickém testování dvoufaktorové autentizace se softwarem OpenVPN a strongSwan.

Obrázek 5.1: *USB token Aladdin eToken PRO 32k [21]*

Tabulka 5.1 zobrazuje základní technické specifikace tohoto USB tokenu. Kompletní technické specifikace se nacházejí v příloze B.

Tabulka 5.1: *Technické specifikace Aladdin eToken PRO 32k [22]*

Technické specifikace	
Podporované operační systémy	**Windows:** Windows 8.1 (32-bit a 64-bit), Windows Server 2012 R2 a všechny předešlé verze. **Mac:** Mac OS X 10.9 (Mavericks) a všechny předešlé verze. **Linux:** Red Hat, Ubuntu, Debian, SUSE, CentOS a Fedora.
API a standartní podpora	PKCS#11 V2.20, MS CryptoAPI a CNG (CSP, KSP), Mac Keychain (TokenD), Java Card 2.2.2, GlobalPlatform Verze 2.1.1, PC/SC, X.509 v3, SSL v3, IPSec/IKE
Bezpečnostní algoritmy	RSA 1024-bit / 2048-bit, DES, 3DES, SHA1
Rozměry	52 x 16 x 8 mm
USB konektor	USB typ A, podpora USB 1.1 a 2.0
Paměť tokenu	32 768 kB
Délka uchování dat	Nejméně 10 let
Počet přepisů paměti	Nejméně 500,000x

USB token eToken PRO je nabízen také v několika dalších variantách s rozdílnou pamětí. USB eToken PRO s velikostí paměti 32 kB a 64 kB používá operační systém Siemens CardOS a token s pamětí 72 kB používá operační systém Java OS.

V následujících podkapitolách je pospán postup instalace a inicializace USB tokenu Aladdin eToken PRO 32k, základní práce s tokenem a jak na něj ukládat klíče a certifikáty pomocí nainstalovaných nástrojů.

5.1 Instalace na OS Ubuntu

Tato kapitola je zaměřena na několik způsobů jak zprovoznit a pracovat s testovaným USB tokenem na operačním systému Ubuntu 14.04 LTS.

5.1.1 OpenSC – nástroje a knihovny pro čipové karty

OpenSC poskytuje sadu knihoven a nástrojů pro práci s čipovými kartami. Hlavní zaměření OpenSC je na karty a tokeny, které podporují kryptografické operace. Implementuje API PKCS#11 a na kartě implementuje standard PKCS#15.OpenSC tedy umožní přístup aplikací k uvedenému technickému zařízení pomocí rozhraní PKCS#15.

- **PKCS#11** – rozhraní pro kryptografický token (Cryptoki). Jedná se o API popisující obecně použitelné rozhraní pro kryptografické tokeny.
- **PKCS#15** – standard přístupu ke kryptografickému tokenu (struktura souborů). Tento standard popisuje, jakým způsobem uživatelé povolují aplikacím pracovat s tokenem, nezávisle na implementaci PKCS#11 nebo jiného API.

OpenSC se skládá z dalších podprojektů, které mohou být použity nezávisle na sobě a bez OpenSC. Pro zprovoznění našeho tokenu ovšem nestačí nainstalovat pouze OpenSC. Po vložení tokenu do USB portu jej sice operační systém Ubuntu detekuje (správnou detekci lze ověřit v seznamu připojených USB zařízení - příkaz `lsusb`), ale pro práci s tokenem a správnou funkci potřebujeme ještě nainstalovat další balíky. Mezi nejdůležitější patří OpenCT, který poskytuje ovladače pro tokeny a čipové karty. Právě OpenCT (tzv. middleware) využijeme pro komunikaci s tokenem. Balíčky nainstalujeme přes klasickou správu balíku pomocí příkazu `apt-get install` nebo manuální kompilací ze zdrojových kódů. Názvy balíků se mohou lišit dle použité verze operačního systému. [29] Níže lze vidět přehled potřebných balíků k instalaci:

- **opensc** – poskytuje sadu knihoven a nástrojů pro práci s čipovými kartami a tokeny.
- **openct** – implementuje ovladače čipových karet a tokenů.
- **libpcsclite1** – tento balík již bývá v OS nainstalován. Software (tzv. middleware) jedná se o knihovny pro přístup k čipové kartě nebo tokenu využívající PC/SC.
- **pcscd** – démon pro přístup k čipové kartě nebo tokenu pomocí PC/SC.
- **libccid** – PC/SC ovladač pro USB CCID čipové karty a tokeny. Tento balík je potřebný pro komunikaci prostřednictvím PC/SC Lite správce zdrojů (démon pcscd), tento balík je většinou nainstalován již s instalací pcscd.

- **pcsc-tools** – tento balík není pro funkci tokenu nutný, ale doporučuji jeho instalaci. Obsahuje některé užitečné nástroje pro práci s čipovými kartami, tokeny a PC/SC.

Spolu s OpenSC se nainstalují nástroje `pkcs15-tool`, `pkcs15-init` a `pkcs11-tool`, které využijeme pro inicializaci a práci s tokenem. OpenSC také nainstaluje modul `opensc-pkcs11.so` sloužící pro komunikaci s tokenem a čipovými kartami.

Když jsou všechny balíky nainstalovány, musí se spustit služby openct a pcscd. Případně že již běží, musí být restartovány.

```
/etc/init.d/openct start
/etc/init.d/pcscd start
```

Poté můžeme vyzkoušet utilitu `openct-tool` nebo `opensc-tool` a zkusit identifikovat vložený token do USB portu.

```
openct-tool list
0 Aladdin eToken PRO 32K
```

Identifikování pomocí utility opensc-tool.

```
opensc-tool --list-readers
#Detected readers (pcsc)
Nr.  Card  Features   Name
0    Yes             Aladdin eToken PRO 32 00 00
```

Také můžeme použít nainstalovaný nástroj `pcsc_scan` z balíku pcsc-tools, který detekuje aktuálně připojené zařízení, a v případě nepřítomnosti zařízení ihned vidíme změnu ve výpisu. Kompletní výpis je přiložen v příloze C.

Nyní již k inicializaci tokenu. Jako první lze vypsat všechny objekty obsažené na tokenu pomocí příkazu, který slouží pro práci s datovými strukturami na tokenech.

```
pkcs15-tool --dump
```

Pokud neobdržíme seznam objektů ale výpis `PKCS#15 initialization failed: Unsupported card`. Tento výpis znamená, že token ještě nebyl inicializován. (nejedná se nutně o nepodporovaný token). Pokud token obsahuje objekty, je možno je smazat a token dostat do neinicializovaného stavu příkazem `pkcs15-init --erase-card`.

Nyní je potřeba na tokenu založit datové struktury a přiřadit ho ke konkrétnímu uživateli (nastavit jeho PIN a PUK kód). Dále je ještě možnost nastavit PIN pro správce tzv. SOPIN (Security Officer PIN), bez kterého nelze provádět některé operace. Doporučuji tento PIN kód také vytvořit, na testovaném USB tokenu Aladdin eToken PRO 32k se bez SOPIN kódu nepodařilo úspěšně vytvořit datovou strukturu.

K inicializaci a vytvoření uživatele slouží nástroj `pkcs15-init`, pomocí kterého provedeme počáteční vytvoření struktury. Prvním příkazem vytvoříme datovou strukturu a budeme požádáni o zadání SOPIN kódu.

```
pkcs15-init --create-pkcs15
```

Následujícím příkazem vytvoříme uživatele s popiskem „Adam“ a příslušným ID číslem 01. Poté musíme zadat PIN a PUK kód. Pokud token používá více lidí, můžeme nastavit více PIN kódů. Správně zadaný PIN umožňuje přístup k objektům, které jsou identifikovány stejným ID číslem `auth-id`.

```
pkcs15-init --store-pin --auth-id 01 --label "Adam"
```

Nyní na token uložíme již dříve vygenerovaný klientský klíč `klient.key` a certifikát `klient.crt`. Následující příkaz na token uloží náš soukromý klíč, přiřadí ho k uživateli „Adam“ dle parametru `auth-id` a nastaví soukromému klíči ID číslo 45.

```
pkcs15-init --store-private-key klient.key --auth-id 01 --id 45
```

Příkaz pro uložení certifikátu je velice podobný, lze také například přidat parametr, v jakém formátu se má klíč nebo certifikát uložit.

```
pkcs15-init --store-certificate klient.crt --auth-id 01 --id 45
```

Pomocí nástroje `pkcs15-init` je také možné vygenerovat RSA klíče pomocí vnitřního generátoru klíčů na tokenu.

```
pkcs15-init --generate-key rsa/2048 --auth-id 01
```

Tímto je inicializace pomocí nástroje `pkcs15-init` dokončena. Na tokenu jsou uloženy požadované soubory a lze jej začít používat k našim účelům. [30] Veškeré funkce lze vypsat pomocí příkazu `pkcs15-init help`. Uložené objekty na tokenu vypíšeme příkazem `pkcs15-tool --dump`, kompletní výpis lze nalézt v příloze D. Výhodou OpenSC a inicializací pomocí nástroje `pkcs15` je, že podporuje spoustu čipových karet a USB tokenů, které jsou zveřejněny na stránkách projektu OpenSC [29].

Inicializace a práce s tokenem pomocí nástroje `pkcs11-tool` je velice obdobná. Zde nevytváříme strukturu dle standardu `pkcs15`, ale provedeme inicializaci s využitím `pkcs11` modulu. Veškeré funkce lze vypsat pomocí příkazu `pkcs11-tool help`. U každého příkazu je povinný parametr `--module`, kterým vybíráme námi požadovaný `pkcs11.so` modul. Níže lze vidět ukázku takovéhoto příkazu, kdy za zadaným modulem následují ostatní parametry v tomto případě parametr `--list-objects` vypíše objekty uložené na tokenu.

```
pkcs11-tool --module /usr/lib64/libeTPkcs11.so --list-objects
```

Důležité je také zmínit, že s inicializovaným tokenem pomocí `pkcs15` a modulu `opensc-pkcs11.so` nelze například pracovat se softwarem XCA (více v podkapitole 5.1.3). V případě, že chceme používat software XCA, je nutné provést inicializaci pomocí `pkcs11` a využívat k tokenu nainstalovaný modul z podkapitoly 5.1.2 na následující straně.

5.1.2 Instalace softwaru eToken PKI Client

V této podkapitole se věnuji instalaci softwaru eToken PKI Client, jedná se o takzvaný „middleware“, který poskytuje PKCS#11 modul (knihovnu). EToken PKI Client umožňuje integraci s různými bezpečnostními aplikacemi a realizaci dvoufaktorové autentizace pomocí standardních certifikátů. Bohužel poslední verze aplikace eToken PKI Client 5.00 pro Ubuntu vyšla v roce 2009 a je udávána podpora pouze pro Ubuntu 8.04 (32-bit) a 9.04 (32-bit) a další starší distribuce operačního systému Linux. Na testovaném Ubuntu 14.04 LTS (32-bit a 64-bit) nefunguje grafické prostředí klienta, ovšem z instalace softwaru využijeme nainstalovaný modul `libeTPkcs11.so`, který lze bez problému využít v aplikacích pro práci s tokenem.

Na operačním systému opět doporučuji mít nainstalován balík `pcsc-tools` a v případě, že chceme využívat nástroj `pkcs11-tool`, také balík `opensc`. Je nutné mít nainstalován balík `libpcsclite1`, balíky `pcscd` a `libccid` se zavedou během instalace softwaru PKI Client. Další doporučené balíky jsou `libqt4-core` a `libqt4-gui` (balíky GUI runtime knihovny).

Instalační balík softwaru PKI Client ve formátu .deb je dostupný pouze na 32 bitovou verzi Ubuntu, na 64 bitovou verzi musíme balík převést z .rpm souboru (formátu) pomocí nástroje `alien`. Příkaz převodu a následnou instalaci lze vidět níže.

```
alien pkiclient-5.00.28-0.x86_64.rpm
dpkg –i pkiclient_5.00.28-1_amd64.deb
```

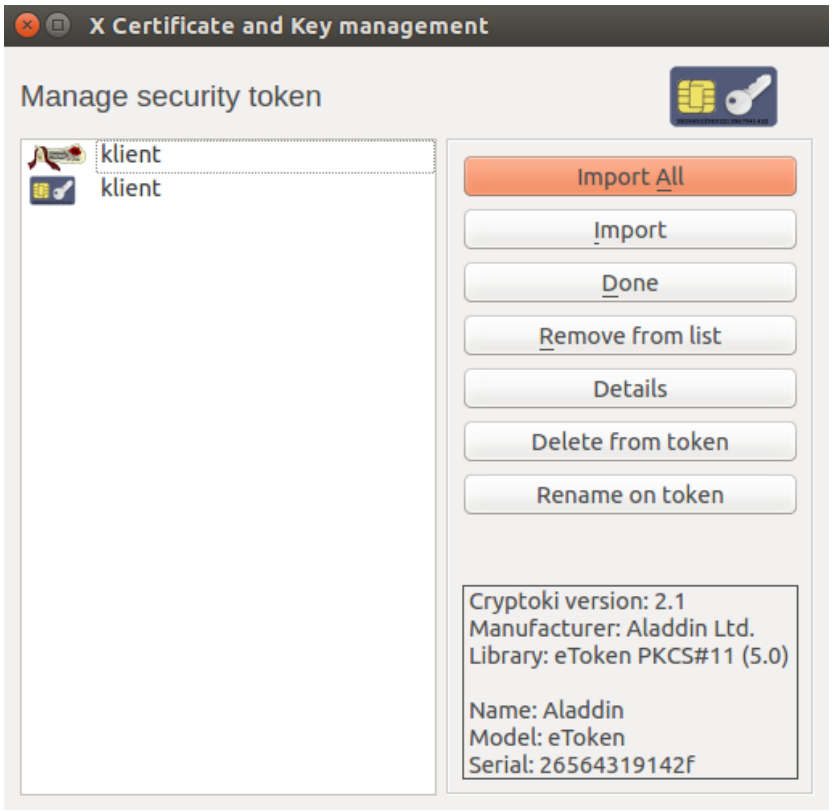
Dále je potřeba nainstalovat balík `libhal1`, který poskytuje jednoduché rozhraní (hardwarovou vrstvu) pro připojení nových zařízení do počítače. Tento balík je nutný pro funkčnost modulu `libeTPkcs11.so` nainstalovaného během instalace softwaru PKI Client. Na 32 bitové verzi Ubuntu je po instalaci balíku `libhal1` již vše funkční a s tokenem lze korektně pracovat (pro načtení tokenu se využívá ovladač AKS ifdh0 nainstalovaný během instalace eToken PKI client). Ovšem na 64 bitové verzi Ubuntu ovladač AKS ifdh0 token nenačetl, je potřeba nainstalovat ještě balík `openct`, který nám toto zajistí.

Společnost SafeNet v roce 2011 vydala software SafeNet Authentication Client 8.1, kde udává podporu pro Aladdin eToken PRO 32k pouze na operačním systému Ubuntu 10.04 (32 a 64 bitové verze) založené na verzi jádra 2.6. Během testováni na OS Ubuntu 14.04 lze po instalaci tohoto klienta opět využívat modul pkcs11. V roce 2014 byla vydána verze SafeNet Authentication Client 8.3, v dokumentaci tohoto softwaru je udána již podpora pro OS Ubuntu 13.10 a 14.04. Pro přístup k softwaru je ovšem potřeba mít u společnosti SafeNet vytvořen účet a tento software zakoupit.

Tabulku všech použitých balíků, softwaru a jejich verzí během instalace tokenu lze nalézt v příloze E. Během dalšího testování a práce s USB tokenem byl využíván modul `libeTPkcs11.so` od nainstalovaného softwaru eToken PKI Client.

5.1.3 Inicializace tokenu pomocí softwaru XCA

XCA neumí pracovat pouze s klíči a certifikáty, které jsou uloženy v počítači, ale podporuje také čipové karty a USB tokeny. K tomuto je potřeba mít nainstalovaný software, který poskytuje PKCS#11 modul (knihovnu), viz předchozí kapitola. Tento PKCS#11 modul je v softwaru XCA nutno zaregistrovat. V menu vybereme položku „File“ a následně „Options“. V otevřeném formuláři v sekci „PKCS#11 provider“ přidáme pomocí tlačítka Add modul `libeTPkcs11.so` z umístění, kde se nainstaloval. Můžeme také využít tlačítko Search a XCA nechat vyhledat modul. Nyní potvrdíme tlačítkem „OK“, tímto se nám v menu zpřístupní položka „Token“ pro práci s naším USB tokenem. V této položce lze inicializovat token, nastavit PIN a také SOPIN tokenu nebo otevřít formulář pro správu tokenu, který je možné vidět na obrázku 5.2.



Obrázek 5.2: *Formulář pro správu tokenu v XCA*

V případě importu soukromého klíče na token stačí na klíč kliknout pravým tlačítkem a zvolit „Store on Security token“. Při importu certifikátu na něj klikneme pravým tlačítkem, vybereme možnost „Export“ a následně „Security token“. Budeme vyzváni k vyplnění PIN kódu, který je na tokenu nastaven. Tímto jsou požadované soubory úspěšně uloženy na token.

5.2 Instalace na OS Windows

Pro inicializaci a práci s USB tokenem na operačním systému Windows nainstalujeme software, který jsme použili také na OS Ubuntu. Jedná se o software eToken PKI Client a XCA pro správu certifikátů.

5.2.1 Instalace softwaru eToken PKI Client

Instalaci softwaru zahájíme spuštěním pomocí příslušného EXE souboru, následně nás instalací provede průvodce. Nainstalován byl eToken PKI Client verze 5.1 SP1 na operačním systému Windows 7, software lze stáhnout zde [31]. Během instalace se do systému zavedou ovladače tokenu, token je pod systémem Windows 7 plně funkční včetně grafického prostředí softwaru eToken PKI Client, pomocí kterého lze s tokenem provádět základní operace. Hlavní okno grafického prostředí softwaru eToken PKI Client je na obrázku 5.3.

Obrázek 5.3: *Software eToken PKI Client*

5.2.2 Inicializace tokenu pomocí softwaru XCA

Na operační systém Windows lze také nainstalovat software XCA pro práci s certifikáty a klíči. Postup zprovoznění tokenu je stejný jako v podkapitole 5.1.3 pod OS Ubuntu. Zde je nutno zaregistrovat PKCS#11 DDL knihovny eTPKCS11.dll a eToken.dll, které se nainstalovaly s eToken PKI Clientem. Základní umístění knihoven je v adresáři `C:/Windows/System32`. Software XCA lze stáhnout z domovské stránky zde [28].

6 Software OpenVPN

V následující kapitole je popsán celkový průběh testování softwaru OpenVPN (verze 2.3.2), návrh a řešení dvoufaktorové autentizace. Na úvod je zde popsána instalace a zprovoznění OpenVPN, popis základní konfigurace a konfiguračních souborů. Jak jsme se již dříve mohli dočíst OpenVPN nabízí několik typů autentizace a to pomocí sdílených klíčů, s využitím certifikátů nebo pomocí uživatelského jména a hesla. Popsány jsou všechny tyto možnosti, ale primární zaměření je zde na využití uživatelského jména a hesla, certifikátů a jejich následné umístění na USB tokenu. Po instalaci a seznámení je již popsán postup řešení pro dvoufaktorovou autentizaci. Celý postup a návrh řešení je soustředěn na OS Ubuntu (verze 14.04 LTS), ovšem je zde také popsáno řešení OpenVPN na OS Windows a následně popsány rozdíly v porovnání s OpenVPN na OS Ubuntu.

6.1 Instalace OpenVPN

V následující podkapitole jsou popsány různé způsoby instalace softwaru OpenVPN. Jako první je uvedena instalace na OS Ubuntu a následně také na OS Windows.

6.1.1 Instalace na Ubuntu

Jako první a nejednoduší způsob je instalace OpenVPN přes klasickou správu balíku, pomocí následujícího příkazu.

```
apt-get install openvpn
```

Pomocí tohoto příkazu se nám nainstaluje software OpenVPN a všechny potřebné balíky pro práci automaticky. Dobré je si zjistit nainstalovanou verzi OpenVPN pomocí následujícího příkazu.

```
openvpn version              #nebo openvpn --version -a
```

Další možností je manuální instalace pomocí kompilace zdrojových kódů. Kdy si stáhneme potřebný soubor ze stránek výrobce [23]. Nutno podotknout, že při instalaci pomocí správy balíků bývají repositáře někdy zastaralé a na stránkách výrobce je k dispozici už novější verze software. Níže lze vidět rozbalení archivu pomocí prvního příkazu a přepnutí v terminálu do rozbalené složky.

```
tar xzf openvpn-2.3.2.tar.gz
cd openvpn-2.3.2
```

Některé programy mohou pro kompilaci vyžadovat další knihovny a instalační balíky. Po rozbalení by se ve složce měl nacházet soubor INSTALL s návodem k instalaci. V rozbalené složce zadáme pro kompilaci následující příkazy.

```
./configure
make
```

Nyní již provedeme instalaci. Při použití prvního příkazu vytvoříme DEB balík, který můžeme později odebrat přes správce balíků. V případě druhého příkazu instalované soubory nepůjdou odebrat přes správce balíků, jediná možnost je příkazem `sudo make uninstall`.

```
checkinstall -D -y --install
```

```
make install
```

Užitečné je také použít příkaz, který je napsán níže. Tento příkaz nám vypíše knihovny a balíky na kterých závisí správná funkce software OpenVPN, které je důležité nainstalovat před manuální instalací pomocí kompilace zdrojových kódů. Mezi požadovanými balíky jsou například `openssl`, `libpam0g`, `libssl1.0.0`, `liblzo2-2`, `libc6` a další. V tomto případě instalace také závisí na distribuci operačního systému a aktuálně dostupných balíčcích (názvy mohou být rozdílné).

```
apt-cache depends openvpn
```

6.1.2 Instalace na Windows

Při instalace OpenVPN na OS Windows stáhneme instalační soubor ze stránek výrobce zde [23]. Vybereme si požadovanou 32-bit nebo 64-bit verzi. Podporován je Windows XP a novější verze Windows. Poté již stačí spustit .exe soubor a instalací nás provede instalační průvodce. Během instalace máme možnost, jaké komponenty chceme nainstalovat. Například OpenSSL utilitu pro generování statických klíčů, OpenVPN RSA certifikační management, jednoduché grafické rozhraní OpenVPN GUI a další. Následně stačí zvolit umístění, kde se OpenVPN nainstaluje. Po úspěšné instalaci již můžeme spustit grafické rozhraní pro připojení k určitému serveru. Pro veškeré informace doporučuji dokumentaci [24].

6.2 Základní konfigurace

V následující podkapitole je popsána základní konfigurace, konfigurační soubory a spuštění samotného OpenVPN, zaměřeného na systém Ubuntu a poté porovnán s Windows platformou.

6.2.1 Konfigurace na OS Ubuntu

Po instalaci se nám vytvoří adresář `/etc/openvpn`, kde nejčastěji umísťujeme konfigurační soubory, ale není to podmínkou. Pro stranu serveru i klienta se instaluje stejný software. Role serveru a klienta je rozlišena až v konfiguračním souboru. Nyní následují konfigurace se základními příkazy. Jedná se o základní spojení dvou stran zabezpečeno pomocí statického (sdíleného) klíče. Na počítači, který bude v roli serveru, vygenerujeme klíč pro symetrické šifrování a dešifrování. Klíč vygenerujeme příkazem níže. Zabezpečeným způsobem přenesme klíč na klientský počítač.

```
openvpn -genkey -secret /etc/openvpn/klic.key
```

Konfigurační soubory a vygenerovaný statický klíč jsou umístěny v `/etc/openvpn`. Na další straně lze vidět konfiguraci na straně serveru. Jako první si vybereme virtuální síťové rozhraní,

které musí být na obou koncích stejné (TUN nebo TAP). Určíme si napevno virtuální adresy tunelu. Následně si vybereme, zda bude vytvořen TCP nebo UDP tunel, požadovaný port, šifrovací algoritmus, hashovací funkci. Zadáme cestu k našemu klíči. Z bezpečnostních důvodů je vždy lepší spouštět program bez administrátorských práv. Práva procesu OpenVPN můžeme snížit po jeho inicializaci tak, aby běžel jako uživatel „nobody“. Pomocí parametru `user nobody` a `group nogroup` tedy zredukujeme po spuštění přístupová práva OpenVPN. Na Windows platformě tyto parametry nejsou podporovány. Parametr verb nastaví požadovanou úroveň logování.

Konfigurační soubor serveru /etc/openvpn/server.conf:

```
dev tun                          #výběr rozhraní
ifconfig 10.0.0.1 10.0.0.2       #IP adresa serveru a klienta
proto udp                        #výběr UDP nebo TCP
port 1194                        #výběr portu
auth SHA256                      #hashovací funkce
cipher AES-256-CBC               #šifrovací algoritmus
secret /etc/openvpn/klic.key     #nastavení statického klíče
copm-lzo                         #zapnutí komprese
user nobody                      #snížení privilegii po spuštění
group nogroup                    #snížení privilegii po spuštění
verb 3                           #úroveň logování
```

Jedná se jen o základní parametry ukázkového příkladu pro seznámení s konfiguračními soubory, pro veškeré možnosti konfigurace a popis parametrů doporučuji dokumentaci na stránkách OpenVPN [24]. Nastavení množství zapisovaných hlášení (úroveň logování) se konfiguruje nejčastěji následujícími úrovněmi. Úrovně 3 a 4 při běžném používání již poskytují dostatek informací. Informace si také můžeme nechat vypisovat do logovacího souboru.

- 0 - tiché s výjimkou velmi závažných chyb.
- 1 - většinou tiché, zobrazuje i nezávažné síťové chyby.
- 3 - střední úroveň, dobrá pro normální použití.
- 4 - použití pro obecné informace.
- 5 a 6 - může pomoct při hledání problému při spojení.
- 9 - vypisuje vše, dobré pro odhalování chyb.

Na další straně lze vidět část konfigurace klienta. Veškerá konfigurace je totožná se stranou serveru až na dva řádky (vybraná rozhraní, šifry atd. musí být stejné na obou koncích). První řádek `remote` nám udává IP adresu nebo Hostname serveru, ke kterému se připojujme, jedná se o jeho fyzickou adresu. Musíme být schopni server dosáhnout pomocí pingu. Pomocí druhého řádku nastavíme virtuální IP adresy na rozhraní TUN, první klienta a poté serveru.

Konfigurační soubor klienta /etc/openvpn/klient.conf:

```
remote 192.168.1.108            #IP adresa serveru (fyzická)
ifconfig 10.0.0.2 10.0.0.1      #IP adresa klienta a serveru
```

Tímto je konfigurace obou stran serveru i klienta hotova. Kompletní konfigurace lze nalézt v příloze F. Při konfiguraci jsou také užitečné následující dva příkazy pro vypsání nabízených šifrovacích algoritmů a hashovacích funkcí.

```
openvpn --show-ciphers
openvpn --show-digests
```

Nyní již stačí pouze spustit službu OpenVPN. Při spuštění OpenVPN se můžeme setkat s několika způsoby. První dva příkazy nám spouští konkrétní konfigurační soubor serveru nebo klienta, předpokládáme, že se nacházíme v adresáři s těmito soubory. Po zadání jednoho z těchto dvou příkazu se nám v terminálu zobrazí stavové informace a průběh spojení (záleží na nastavení parametru verb pro logování), při zavření okna terminálu se služba OpenVPN ukončí. Posledním příkazem trvale spustíme službu OpenVPN.

```
openvpn server.conf
openvpn --config server.vpn
service openvpn start
```

Pro ověření funkčnosti použijeme ping na virtuální adresu serveru - `ping 10.0.0.1`. Při připojování k serveru v roli klienta je také možnost využít například grafický doplněk do správce sítě, těmto dalším možnostem a řešením je krátce věnován konec této kapitoly.

6.2.2 Konfigurace na OS Windows

OpenVPN se nám nejčastěji nainstaluje do `C:\Program Files\OpenVPN`. V této cestě se nám vytvoří adresář `C:\Program Files\OpenVPN\config`, kde se umísťují veškeré konfigurační soubory. Pro server i klienta se instaluje stejný software. Stejně jako u platformy Linux je konfigurace prováděna pomocí konfiguračních souborů (skriptů) a až zde je rozlišeno, zda se jedná o funkci klienta nebo serveru. Konfigurační soubor, který si můžeme nazvat libovolně má vždy koncovku .ovpn. Obsah konfiguračního souboru je stejný jak pro operační systém Windows, tak pro Linux. Jediný rozdíl v konfiguračním souboru je v oddělovači souborů při zadáváni cesty, který je na Windows „\“, ale na platformě Linux je „/“.

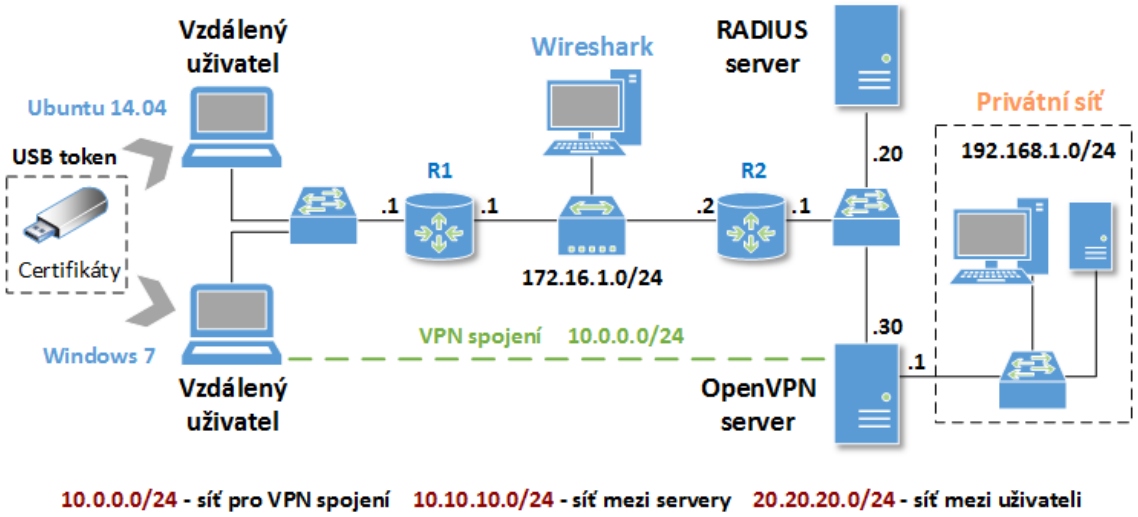
```
# systém Linux
secret /etc/openvpn/klic.key
# systém Windows
secret "C:\\Program Files\\OpenVPN\\config\\klic.key"
```

Co se týče spuštění OpenVPN stačí v tomto případě kliknout pravým tlačítkem na konfigurační soubor a zvolit možnost „Start OpenVPN on this config file“ nebo z příkazového

řádku spustit následovně `openvpn [konfigurační soubor]`. V případě že se připojujeme, jako klient k serveru je dobré využít grafické rozhraní OpenVPN GUI, kdy pouze klikneme pravým tlačítkem myši na ikonu v oznamovací oblasti, vyvoláme místní nabídku, zvolíme server a potvrdíme možností „connect“.

6.3 Návrh řešení - dvoufaktorová autentizace

V následující podkapitole je testována dvoufaktorová autentizace pomocí řešení OpenVPN na navrhnutém zapojení v laboratorních podmínkách. Je zde popsán kompletní návod na zprovoznění tohoto navrhnutého řešení. Během testování bude použita vlastní CA a certifikáty, které jsme vygenerovali již dříve (viz kapitola 4). Použitý USB token již obsahuje klientské certifikáty, které jsme na něj umístili pomocí softwaru XCA v kapitole 5.1.3. Na serverech je použit OS Ubuntu 14.04 (64-bit verze), na klientských PC je poté použit OS Ubuntu 14.04 a také odzkoušeno připojení z OS Windows 7. Nakonec je ověřena celková funkčnost tohoto řešení. Na obrázku 6.1 lze vidět schéma zapojení.



Obrázek 6.1: *Navrhnuté schéma zapojení*

Výše na schématu zapojení vidíme jednotlivé zařízení a propojení mezi nimi. Nachází se zde OpenVPN server, který zpřístupňuje připojeným vzdáleným uživatelům privátní síť. Jedná se o připojení VPN pro vzdálený přístup (zařízení-síť), pomocí OpenVPN je možno nakonfigurovat také další typy spojení. Klienti jsou autentizováni pomocí digitálních certifikátů a také pomocí uživatelského jména a hesla. Při vytváření VPN spojení OpenVPN server zasílá požadavek na RADIUS server pro ověření uživatelského jména a hesla, které jsou uloženy v MySQL databázi. Autentizace dle uživatelského jména a hesla je prováděna tedy externě na RADIUS serveru. Autentizaci pomocí digitálních certifikátů již obstarává samotný OpenVPN server. Na trase mezi serverem a klienty jsou dva směrovače R1 a R2, které provádí směrování mezi sítěmi. Dále jsou již popsány jednotlivé kroky konfigurace.

6.3.1 Konfigurace OpenVPN serveru

Následně je popsán postup konfigurace OpenVPN serveru. Předpokládáme, že konfigurační soubor a další potřebné soubory jsou umístěny v `/etc/openvpn`. Pomocí prvního parametru povolíme protokol TLS, server bude zároveň sloužit i pro výměnu certifikátů. Dále specifikujeme „mód server“, tento mód je určen pro podporu až stovek uživatelů (záleží na výkonnosti hardware), je zde nutné použití SSL/TLS autentizace. Virtuální IP adresy tunelu jsou přidělovány dynamicky z rozsahu pomocí třetího parametru `server`. Parametrem `push` je klientům nabízena privátní podsíť skrz vytvořené VPN spojení. Dále již následuje výběr virtuálního síťového rozhraní (tun), vytvoření UDP tunelu, výběr požadovaného portu. Parametr `keepalive` slouží k ověřování, že druhá strana stále reaguje (každých 10 sekund ping, do 120 sekund neobdržena odpověď – spojení přerušeno).

```
tls-server
mode server
server 10.0.0.0 255.255.255.0
push "route 192.168.1.0 255.255.255.0"
dev tun
proto udp
port 1194
keepalive 10 120
```

Následujícími dvěma parametry `cipher` a `auth` vybereme šifrovací algoritmus (AES-256-CBC) a hashovací funkci SHA1. Dále zadáme cesty k potřebným souborům – pro kořenový certifikát CA, certifikát serveru, soukromý klíč serveru a soubor Diffie-Hellmanova algoritmu.

```
cipher AES-256-CBC
auth SHA1
ca ca.crt
cert server.crt
key server.key
dh dh2048.pem
tls-auth ta.key 0
```

Poslední parametr `tls-auth` přidává dodatečnou HMAC autentizaci v TLS kontrolním kanálu pro ochranu proti DoS útokům a UDP port floodingu. Statický klíč vygenerujeme například pomocí příkazu `openvpn --genkey --secret ta.key`. Tento klíč by měl být uchován v tajnosti. Na straně serveru se u tohoto parametru nastaví hodnota na „0“, na straně klienta poté na hodnotu „1“.

Dále pomocí příkazu `comp-lzo` zapneme kompresi při přenosu skrze VPN. Parametrem `verb` nastavíme úroveň logování a pomocí `max-clients` řekneme, kolik může být maximálně zároveň připojeno klientů. Pomocí parametru `user nobody` a `group nogroup` zredukujeme po spuštění přístupová práva OpenVPN. Volby `persist-tun` a `persist-key` se pokusí zabránit přístupu k určitým prostředkům po restartu, které již nemusí být k dispozici kvůli změně (snížení) přístupových práv.

```
comp-lzo
verb 3
max-clients 10
user nobody
group nogroup
persist-tun
persist-key
```

Tímto je server nastaven a již funkční, ale autentizace uživatelů by probíhala pouze na základě certifikátů. Pro to, aby bylo po uživatelích vyžadováno uživatelské jméno a heslo, je potřeba v konfiguračním souboru využívat doplňky (tzv. pluginy) nebo odkazovat na vytvořené skripty. Níže lze vidět nastavení doplňku v konfiguračním souboru, tento doplněk je nainstalován již během instalace OpenVPN.

```
plugin /usr/lib/openvpn/openvpn-plugin-auth-pam.so login
```

Při použití tohoto doplňku je nutné, aby na serveru byly uživatelské účty a hesla vytvořeny lokálně. Pokud bychom chtěli autentizaci provádět pouze s uživatelským jménem a heslem, lze pomocí parametru `client-cert-not-required` vypnout autentizaci pomocí certifikátů. Pro autentizaci na základě uživatelského jména a hesla ovšem využijeme FreeRADIUS server, který poskytne komplexnější a efektivnější řešení. Jako první musíme nainstalovat OpenVPN RADIUS autentizační modul pomocí příkazu níže.

```
apt-get install openvpn-auth-radius
```

Během instalace se nám vytvoří konfigurační soubor `radiusplugin.cnf` a autentizační modul `radiusplugin.so` v následujících adresářích.

```
/usr/lib/openvpn/radiusplugin.so
/usr/share/doc/openvpn-auth-radius/examples/radiusplugin.cnf
```

Tyto soubory zkopírujeme do adresáře, kde se nachází konfigurace OpenVPN serveru v našem případě `/etc/openvpn`. Do konfiguračního souboru serveru přidáme následujícím řádkem plugin, který nám zajistí propojení s RADIUS serverem.

```
plugin /etc/openvpn/radiusplugin.cnf /etc/openvpn/radiusplugin.so
```

Dále v konfiguračním souboru `radiusplugin.cnf` musíme definovat parametry RADIUS serveru. Jako první musí být nastaven UDP port na 1813 pro účtování a druhý UDP port 1812 slouží pro autentizaci (tyto čísla portů jsou oficiálně přidělané pro RADIUS protokol). Dále zadáme jméno nebo IP adresu RADIUS serveru, kolikrát by měl plugin zaslat požadavek, pokud není žádná odpověď a jak dlouho by měl plugin čekat na odpověď. Jako poslední musíme zadat takzvaný „shared secret“, který je stejně nastaven také na RADIUS serveru.

```
server
{
      acctport=1813
      authport=1812
      name=192.168.1.102
      retry=2
      wait=1
      sharedsecret=openvpn
}
```

Nyní nastavíme síťová rozhraní na serveru, na kterých zapneme IPv4 směrovaní, neboli předávání paketů z jednoho rozhraní na druhé. Otevřeme konfigurační soubor `sysctl.conf`.

```
gedit /etc/sysctl.conf
```

V tomto konfiguračním souboru odkomentujeme následující řádek. Hodnota musí být nastavena na 1, pokud je nastavena na 0, směrování je vypnuto.

```
net.ipv4.ip_forward=1
```

Následně provedeme restart síťové služby nebo použijeme příkaz `sysctl -p`, aby se nastavené změny projevily. Také může být potřeba na serveru nastavit pravidla firewallu například pomocí iptables. Kompletní konfigurace serveru se nachází v příloze G, také doporučuji dokumentaci s popisem všech parametrů [24] [26].

6.3.2 Konfigurace klienta

Nyní je potřeba provést konfiguraci klientské strany. Je zde popsána pouze část konfigurace, totožné parametry se stranou serveru zde nejsou již znovu popsány. Předpokládáme, že konfigurační soubor a další potřebné soubory jsou umístěny v adresáři `/etc/openvpn`. Na další straně jsou již jednotlivé parametry nastavení. Parametr `tls-client` specifikuje, že při výměně certifikátu zařízení vystupuje v roli klienta. Dále musíme použít parametr `pull`, díky kterému klient přijme konfiguraci od serveru. Parametr `remote` nás odkazuje na IP adresu/Hostname serveru, ke kterému se připojíme, jedná se o fyzickou IP adresu. OpenVPN server musí být dosažitelný pomocí pingu.

Další parametr `nobind` zajistí, že klienti se nemusí vázat na specifický lokální port. Aby se zabránilo možnému útoku *„Man-in-the-middle“*, je dobré nastavit nějaké ověření certifikátů serveru klienty. Slouží pro to několik způsobů, například možnost `remote-cert-tls` (na OpenVPN verze 2.0 a níže je to možnost `ns-cert-type`). V nové verzi OpenVPN je také parametr `verify-x509-name`, díky kterému přijmeme připojení pouze v případě, že hostitelovo jméno x.509 je shodné s našim zadaným (v našem případě jméno *„server“*).

```
tls-client
pull
remote 192.168.1.109
nobind
remote-cert-tls server
verify-x509-name server name
```

Níže lze vidět zadání potřebných souborů - kořenový certifikát CA, certifikát klienta, soukromý klíč klienta a dodatečnou TLS autentizaci s nastavením hodnoty na „1“. Parametr `auth-user-pass` zajišťuje autentizaci pomocí uživatelského jména a hesla.

```
ca ca.crt
cert klient.crt
key klient.key
tls-auth ta.key 1
auth-user-pass
```

V tomto případě by se klient již úspěšně připojil k serveru (za předpokladu že na straně serveru je konfigurace hotova včetně funkčního RADIUS serveru). Ovšem klientský certifikát je stále uložen lokálně na uživatelském PC, pro větší bezpečnost jsme jej spolu se soukromým klíčem uložili také na USB token. Důležité je aby ID hodnota certfikátu i privátního klíče na tokenu byla nastavená stejně. Pokud jsou tyto ID hodnoty rozdílné, autentizace a navázání spojení neproběhne úspěšně. USB token vložíme do PC a pomocí následujícího příkazu získáme pkcs11-id certifikátu uloženého na tokenu (výpis příkazu umístěn v příloze G).

```
openvpn --show-pkcs11-ids /usr/lib64/libeTPkcs11.so
```

Jako první musíme zakomentovat nebo smazat parametry `cert` a `key` v konfiguračním souboru (pro bezpečnost také smažeme konkrétní soubory), poté do konfigurace vložíme následující řádky se získaným ID našeho certifikátu.

```
pkcs11-providers /usr/lib64/libeTPkcs11.so
pkcs11-id 'Aladdin\x20Ltd\x2E/eToken/26565946222f/Aladdin/7F558'
```

Nyní při spuštění OpenVPN budeme již vyzvání k vložení USB tokenu do PC a zadání PIN kódu, který chrání token. Kompletní konfiguraci lze nalézt v příloze G. [26]

6.3.3 Konfigurace FreeRADIUS a MySQL databáze

V této podkapitole je popsána konfigurace FreeRADIUS serveru s MySQL databází, která bude využívána pro ukládání uživatelských jmen a hesel. Jako první nainstalujeme pomocí příkazu `apt-get install` následující balíky, které budeme potřebovat.

- **freeradius** – balík obsahující konfigurovatelný RADIUS server.
- **freeradius-mysql** – MySQL modul pro FreeRADIUS server.
- **freeradius-utils** – FreeRADIUS klientské nástroje.
- **mysql-server** – MySQL databázový server.

Po instalaci se nám vytvoří v `/etc/` adresář `/freeradius` a v něm potřebné konfigurační soubory. Pomocí textového editoru otevřeme soubor `radiusd.conf`.

```
gedit /etc/freeradius/radiusd.conf
```

V tomto konfiguračním souboru odkomentujeme text `$INCLUDE sql.conf` pro zahrnutí souboru `sql.conf` (řádek 700). Nyní musíme upravit soubor `default` a v něm odkomentovat řádek 177 v sekci „authorize {}“, řádek 406 v sekci „accounting {}“ a řádek 454 v sekci „session {}“, které obsahují text „sql“.

```
gedit /etc/freeradius/sites-available/default
```

Dále je potřeba v souboru `inner-tunnel` odkomentovat řádek 131 v sekci „authorize {}“ a řádek 255 v sekci „session {}“, které obsahují taktéž text „sql“.

```
/etc/freeradius/sites-available/inner-tunnel
```

Jako poslední musíme v souboru `clients.conf` přidat takzvaného „klienta“ což je náš OpenVPN server, který se bude na FreeRADIUS obracet s požadavky.

```
gedit /etc/freeradius/clients.conf
```

Přidání klienta vypadá následovně. Musí být zadána IP adresa OpenVPN serveru, dále je důležité nastavit tzv. „secret“, který se zadával i v konfiguraci na OpenVPN serveru. [32]

```
client 10.10.10.20 {
      secret      = openvpn
      shortname   = OpenVPNServer
      nastype     = other
}
```

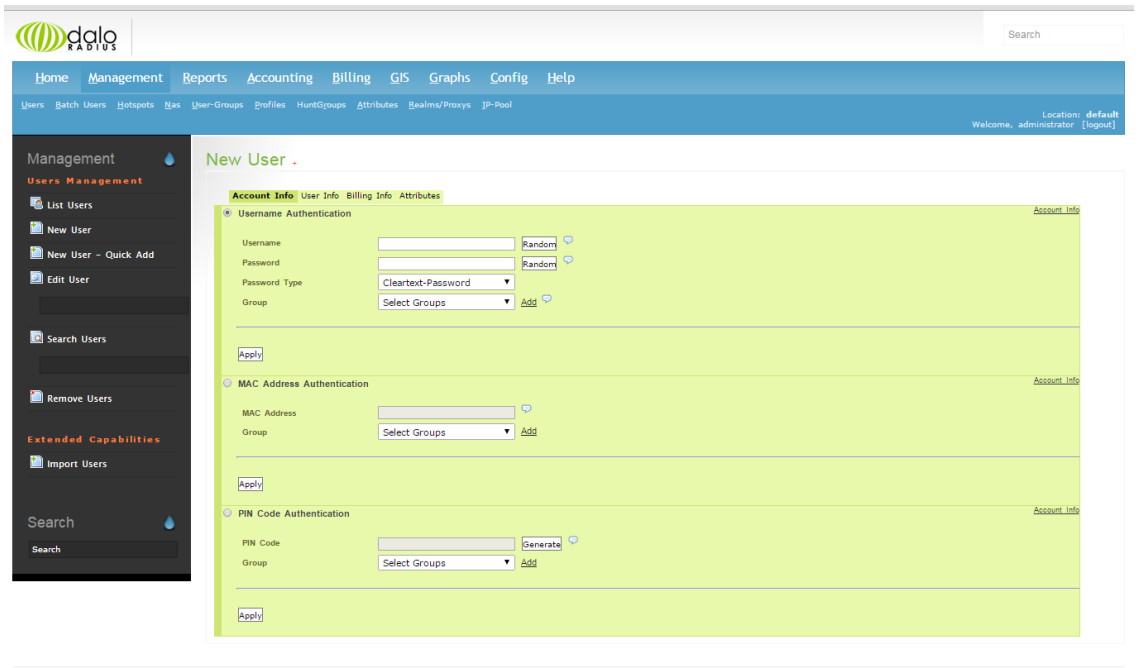
Kompletní návod na konfiguraci MySQL databáze se nachází v příloze H. V případě, že bychom nechtěli využívat MySQL databázi, lze využít obyčejný soubor `users` v adresáři `etc/freeradius`, kde lze přidávat uživatele, ovšem toto řešení není zrovna efektivní. Nakonec je zapotřebí službu FreeRADIUS restartovat, aby se nastavené změny projevily. [33]

6.3.4 Web management daloRADIUS

DaloRADIUS je open source aplikace pro webovou správu RADIUS serveru. Vyskytují se zde například funkce pro správu uživatelů, sledování log souborů, správa účtování, vykreslování grafů a další. Výhodou je právě správa uživatelů a skupin, které bychom jinak museli nastavovat ručně pomocí MySQL konzole v příkazové řádce. Webovou správu DaloRADIUS pro platformu Linux je možné stáhnout [34]. Níže jsou vypsány potřebné balíky pro správnou funkcionalitu aplikace daloRADIUS, které nainstalujeme pomocí klasické správy balíků.

- **php5-mysql** – MySQL modul pro php5, potřebný pro propojení s MySQL databází.
- **php5, php5-gd, php-pear, php-db** – instalace samotného PHP5 a dalších potřebných modulů pro správnou funkci.

Staženou aplikaci daloRADIUS v souboru s příponou .tar.gz pouze rozbalíme pomocí příkazu `tar xzf`. V rozbaleném adresáři se poté nachází veškeré soubory aplikace daloRADIUS. [32] Také je potřeba nastavit práva některých logovacích souboru a adresářů, aby k nim aplikace daloRADIUS měla přístup. Kompletní postup konfigurace a zprovoznění je přiloţen v příloze I. Na obrázku 6.2 je zobrazeno grafické rozhraní aplikace.



Obrázek 6.2: *Aplikace daloRADIUS*

Aplikace daloRADIUS je poté přístupná přes webový prohlížeč, kde zadáme IP adresu RADIUS serveru a samotné umístění adresáře s aplikací daloRADIUS, v našem případě `http://10.10.10.20/daloradius`. Přihlašovací údaje jsou defaultně nastaveny na uživatelské jméno „administrator“ a heslo „radius“, které lze jednoduše změnit přes grafické rozhraní aplikace.

6.3.5 **Ověření funkčnosti**

V následující podkapitole je ověřena funkčnost nakonfigurovaného řešení. Před samotným spuštěním a testováním je potřeba restartovat požadované služby, případně zkontrolovat, zda běží s našimi provedenými změnami. Dále jsou popsány výpisy VPN spojení jak ze strany serveru, tak ze strany klienta.

Po zadání příkazu `openvpn server.conf` pro spuštění OpenVPN je dobré zkontrolovat pomocí příkazu `ifconfig`, zda bylo vytvořeno naše virtuální rozhraní tun0; výpis lze nalézt v příloze J. Následně je možné vidět pouze vybrané části z logu při spuštění a připojení klienta na straně serveru, kompletní výpisy se nachází v příloze K. Mezi prvními informacemi po spuštění lze vidět například načítání doplňků a potřebných souborů. Níže se nachází načítání konfigurace pro doplněk RADIUS a také načítání statického klíče ta.key.

```
Tue Mar 10 12:25:05 2015 RADIUS-PLUGIN: Configfile name:
/etc/openvpn/radiusplugin.cnf.

Tue Mar 10 12:25:05 2015 Control Channel Authentication: using
'ta.key' as a OpenVPN static key file
```

Načítají se zde veškeré naše nastavené parametry - maximální počet připojených klientů, šifrovací algoritmus, hashovací funkce a další.

Nyní lze vidět spuštění virtuálního rozhraní tun0 (TUN/TAP ovladače), zvětšení hodnoty MTU a přiřazení IP adresy virtuálnímu rozhraní. Celý proces úspěšného spuštění serveru končí zprávou Initialization Sequence Completed, následně jsou vypisovány informace o aktivitě na serveru (připojení klientů atd.).

```
Tue Mar 10 12:25:05 2015 TUN/TAP device tun0 opened

Tue Mar 10 12:25:05 2015 /sbin/ip link set dev tun0 up mtu 1500

Tue Mar 10 12:25:05 2015 /sbin/ip addr add dev tun0 local
10.0.0.1 peer 10.0.0.2

Tue Mar 10 12:25:05 2015 Initialization Sequence Completed
```

Nyní přecházíme k navazování komunikace s klientem s IP adresou 20.20.20.10. Vidíme navazovaní spojení a přijetí počátečního paketu od klienta.

```
Tue Mar 10 12:25:24 2015 20.20.20.10:38224 TLS: Initial packet
from [AF_INET]20.20.20.10:38224, sid=cc25f24b 86f111f2
```

Následně je ověřen kořenový certifikát certifikační autority, také lze vidět nastavené parametry certifikátu.

```
Tue Mar 10 12:25:30 2015 20.20.20.10:38224 VERIFY OK: depth=1,
C=CZ, ST=NA, L=Ostrava, O=VSB, OU=MOB, CN=VSB CA, name=DP,
emailAddress=tab0012@vsb.cz
```

Dále je úspěšně ověřen klientský certifikát, níže lze vidět, že se jedná o certifikát s nastaveným parametrem CN=klient.

```
Tue Mar 10 12:25:30 2015 20.20.20.10:38224 VERIFY OK: depth=0,
C=CZ, ST=NA, L=Ostrava, O=VSB, OU=MOB, CN=klient, name=DP,
emailAddress=tab0012@vsb.cz
```

Poté na pozadí následuje spuštění a volání doplňku pro komunikaci s RADIUS serverem, který je potřebný pro autentizaci pomocí uživatelského jména a hesla. Následuje úspěšná autentizace s uživatelským jménem „klient“.

```
Tue Mar 10 12:25:30 2015 RADIUS-PLUGIN: FOREGROUND THREAD:
Auth_user_pass_verify thread started.

Tue Mar 10 12:25:30 2015 20.20.20.10:38224 PLUGIN_CALL: POST
/etc/openvpn/radiusplugin.so/PLUGIN_AUTH_USER_PASS_VERIFY
status=0

Tue Mar 10 12:25:30 2015 20.20.20.10:38224 TLS:
Username/Password authentication succeeded for username 'klient'
```

Následně je vytvářeno sousedské spojení s klientem. Určen šifrovací algoritmus a hashovací funkce pro šifrování a dešifrování dat dle konfigurace. Ze zadaného rozsahu IP adres je určena virtuální IP adresa pro klienta.

```
Tue Mar 10 12:25:30 2015 20.20.20.10:38224 [klient] Peer
Connection Initiated with [AF_INET]20.20.20.10:38224

Tue Mar 10 12:25:30 2015 klient/20.20.20.10:38224 MULTI: Learn:
10.0.0.6 -> klient/20.20.20.10:38224

Tue Mar 10 12:25:30 2015 klient/20.20.20.10:38224 MULTI: primary
virtual IP for klient/20.20.20.10:38224: 10.0.0.6
```

Klient na server zašle zprávu „PUSH_REQUEST“ pro získání konfigurace. Server následně odpoví zprávou „PUSH_REPLY“. Také lze vidět obsah zaslané zprávy „PUSH_REPLY“.

```
Tue Mar 10 12:25:32 2015 klient/20.20.20.10:38224 PUSH: Received
control message: 'PUSH_REQUEST'

Tue Mar 10 12:25:32 2015 klient/20.20.20.10:38224
send_push_reply(): safe_cap=940

Tue Mar 10 12:25:32 2015 klient/20.20.20.10:38224 SENT CONTROL
[klient]: 'PUSH_REPLY,route 192.168.1.0 255.255.255.0,route
10.0.0.1,topology net30,ping 10,ping-restart 120,ifconfig
10.0.0.6 10.0.0.5' (status=1)
```

Kompletní výpis logu ze serveru lze nalézt v příloze K. Na další straně je již rozebrán výpis logu při navazování spojení ze strany klienta.

Na klientské straně spustíme pomocí příkazu `openvpn klient.conf` službu OpenVPN. Níže se nachází vybrané části z logu (verb 3) při sestavování spojení ze strany klienta, kompletní výpisy jsou k dispozici v příloze K.

Jako první, co po nás bude vyžadováno, je zadání uživatelské jména a hesla. Následně se načtou doplňky, v našem případě se načte PKCS#11 modul a statický klíč ta.key.

```
Enter Auth Username:klient

Enter Auth Password:

Tue Mar 10 12:29:49 2015 PKCS#11: Adding PKCS#11 provider
'/usr/lib64/libeTPkcs11.so'

Tue Mar 10 12:29:50 2015 Control Channel Authentication: using
'ta.key' as a OpenVPN static key file
```

Následuje pokus o navázání spojení se serverem s IP adresou 10.10.10.30 a přijetí počátečního paketu od serveru. Ověří se kořenový certifikát certifikační autority.

```
Tue Mar 10 12:29:51 2015 TLS: Initial packet from
[AF_INET]10.10.10.30:1194, sid=d3999531 9d5cada2

Tue Mar 10 12:29:51 2015 VERIFY OK: depth=1, C=CZ, ST=NA,
L=Ostrava, O=VSB, OU=MOB, CN=VSB CA, name=DP,
emailAddress=tab0012@vsb.cz
```

Během navazování spojení je také ověřováno definované použití klíče „Key usage“ a rozšířená použití klíče „Extended Key Usage“. Tyto parametry jsme nastavovali při generování certifikátů, viz kapitola 4.2.

```
Tue Mar 10 12:29:51 2015 Validating certificate key usage

Tue Mar 10 12:29:51 2015 VERIFY KU OK

Tue Mar 10 12:29:51 2015 Validating certificate extended key

Tue Mar 10 12:29:51 2015 VERIFY EKU OK
```

V následujícím výpisu lze vidět ověření jména serveru, které jsme zadávali v konfiguraci parametrem `verify-x509-name server name`, abychom zajistili připojení jen k požadovanému serveru.

```
Tue Mar 10 12:29:51 2015 VERIFY X509NAME OK: C=CZ, ST=NA,
L=Ostrava, O=VSB, OU=MOB, CN=server, name=DP,
emailAddress=tab0012@vsb.cz
```

Dále také proběhlo ověření certifikátu serveru pomocí nastaveného parametru `remote-cert-tls server`.

```
Tue Mar 10 12:29:51 2015 VERIFY OK: depth=0, C=CZ, ST=NA,
L=Ostrava, O=VSB, OU=MOB, CN=server, name=DP,
emailAddress=tab0012@vsb.cz
```

Pokud není token vložen v USB portu, budeme k tomu vyzváni. Dále je po nás vyžadováno heslo pro přístup na USB token, aby autentizace na základě certifikátů (které se nachází na USB tokenu) byla úspěšná.

```
Token-insertion-request|Please insert Aladdin token:

Enter Aladdin token Password:
```

Určí se šifrovací algoritmus a hashovací funkce pro šifrování a dešifrování dat. Následně je vytvářeno sousedské spojení s klientem. Klient na server zašle zprávu „PUSH_REQUEST“ pro získání konfigurace. Server následně odpoví zprávou „PUSH_REPLY“.

```
Tue Mar 10 12:29:59 2015 SENT CONTROL [server]: 'PUSH_REQUEST'
(status=1)

Tue Mar 10 12:30:00 2015 PUSH: Received control message:
'PUSH_REPLY,route 192.168.1.0 255.255.255.0,route
10.0.0.1,topology net30,ping 10, ping-restart 120,ifconfig
10.0.0.6 10.0.0.5'
```

Při získání zprávy „PUSH_REPLY“ je proveden import získaných možností a nastavení, například nastavení časovačů a IP adres. Níže lze vidět určení brány pro klienta, spuštění virtuálního rozhraní tun0 a přiřazení virtuálních IP adres tomuto rozhraní.

```
Tue Mar 10 12:30:00 2015 ROUTE_GATEWAY 20.20.20.1/255.255.255.0
IFACE=eth0 HWADDR=74:d4:35:7c:29:ed

Tue Mar 10 12:30:00 2015 TUN/TAP device tun0 opened

Tue Mar 10 12:30:00 2015 /sbin/ip addr add dev tun0 local
10.0.0.6 peer 10.0.0.5
```

Dále je přidána cesta jak do privátní sítě 192.168.1.0, tak k dosažení virtuální IP adresy 10.0.0.1 serveru přes IP adresu 10.0.0.5. Úspěšné navázání spojení je ukončeno zprávou Initialization Sequence Completed.

```
Tue Mar 10 12:30:00 2015 /sbin/ip route add 192.168.1.0/24 via
10.0.0.5

Tue Mar 10 12:30:00 2015 /sbin/ip route add 10.0.0.1/32 via
10.0.0.5

Tue Mar 10 12:30:00 2015 Initialization Sequence Completed
```

Jako první můžeme zkontrolovat pomocí příkazu `ifconfig`, zda bylo vytvořeno naše virtuální rozhraní tun0. Výpis se nachází v příloze J. Dále také musí být virtuální IP adresa serveru a privátní síť 192.168.1.0 dosažitelná pomocí pingu. Také doporučuji následující kapitolu [35] o způsobu přiřazování IP adres v OpenVPN na oficiálních stránkách.

Nyní se podíváme na zachycenou komunikaci pomocí softwaru Wireshark. Na obrázku 6.3 je zobrazeno několik zpráv při navazování spojení mezi klientem a serverem (nejedná se o všechny zasílané zprávy). Lze vidět IP adresy klienta a serveru. Dále také protokol a o jakou zprávu se jedná. Například zpráva Client Hello zaslána od klienta na server v sobě obsahuje přesnou verzi TLS protokolu, seznam podporovaných šifer, informace o kompresi a další parametry. Obsah zprávy „Client Hello“ je uveden v příloze K.

Source	Destination	Protocol	Length	Info
20.20.20.10	10.10.10.30	TLSv1	423	Client Hello
10.10.10.30	20.20.20.10	OpenVPN	118	MessageType: P_ACK_V1
20.20.20.10	10.10.10.30	OpenVPN	706	MessageType: P_CONTROL_V1 (Message fragment 4)
10.10.10.30	20.20.20.10	TLSv1	159	Server Hello, Certificate, Server Key Exchange
20.20.20.10	10.10.10.30	OpenVPN	170	MessageType: P_ACK_V1
20.20.20.10	10.10.10.30	OpenVPN	210	MessageType: P_CONTROL_V1 (Message fragment 6)
20.20.20.10	10.10.10.30	TLSv1	118	Certificate, Client Key Exchange, Certificate
10.10.10.30	20.20.20.10	OpenVPN	118	MessageType: P_ACK_V1

Obrázek 6.3: *Zachycené zprávy při vytváření VPN spojení*

Na obrázku 6.4 je zobrazena zachycená komunikace mezi OpenVPN a RADIUS serverem. Jedná se o komunikaci, která nastala během připojování klientské stanice s uživatelským jménem „klient“, kdy OpenVPN server pošle požadavek na ověření RADIUS serveru. Lze vidět požadavky a odpovědi pro autentizaci a takzvané účtování (accounting).

Source	Destination	Protocol	Length	Info
10.10.10.30	10.10.10.20	RADIUS	168	Access-Request(1) (id=205, l=126)
10.10.10.20	10.10.10.30	RADIUS	62	Access-Accept(2) (id=205, l=20)
10.10.10.30	10.10.10.20	RADIUS	168	Accounting-Request(4) (id=109, l=126)
10.10.10.20	10.10.10.30	RADIUS	62	Accounting-Response(5) (id=109, l=20)

Obrázek 6.4: *Zachycená komunikace s RADIUS serverem*

Na obrázku 6.5 je zobrazen výřez ze softwaru Wireshark zachycené zprávy Acces-Request. Ve zvýrazněných částech můžeme vidět použité uživatelské jméno „klient“ a uživatelské heslo, které je zašifrováno. Dále také IP adresu, ze které informace přišly. Na tento požadavek přišla kladná odpověď Acces-Accept.

```
⊟ Radius Protocol
    Code: Access-Request (1)
    Packet identifier: 0xcd (205)
    Length: 126
    Authenticator: db4606c3828c031b41fca6eb099c8fb7
    [The response to this request is in frame 12]
  ⊟ Attribute Value Pairs
    ⊟ AVP: l=8 t=User-Name(1): klient
        User-Name: klient
    ⊟ AVP: l=18 t=User-Password(2): Encrypted
        User-Password (encrypted): 3c33da59dbd25adfeb3cc58a0b3fd82b
    ⊞ AVP: l=6 t=NAS-IP-Address(4): 127.0.0.1
    ⊞ AVP: l=6 t=NAS-Port(5): 1
    ⊞ AVP: l=6 t=Service-Type(6): Dialout-Framed-User(5)
    ⊞ AVP: l=13 t=Calling-Station-Id(31): 20.20.20.10
    ⊞ AVP: l=9 t=NAS-Identifier(32): OpenVpn
    ⊞ AVP: l=34 t=Acct-Session-Id(44): C05245A0614729C316D77652C4DCE9BE
```

Obrázek 6.5: *Obsah zprávy Acces-Request na RADIUS*

Na obrázku 6.6 je zachycena šifrovaná komunikace, jedná se o komunikaci mezi klientem a privátní sítí 192.168.1.0 (stanice s IP adresou 192.168.1.10). V tomto případě nelze vidět, o jaký typ komunikace se jedná. Vidíme pouze IP adresu klienta a IP adresu OpenVPN serveru a dále také že se jedná o protokol OpenVPN. Za serverem do privátní sítě je komunikace již nešifrovaná.

Source	Destination	Protocol	Length	Info
20.20.20.10	10.10.10.30	OpenVPN	149	MessageType: P_DATA_V1
10.10.10.30	20.20.20.10	OpenVPN	149	MessageType: P_DATA_V1
20.20.20.10	10.10.10.30	OpenVPN	213	MessageType: P_DATA_V1
10.10.10.30	20.20.20.10	OpenVPN	213	MessageType: P_DATA_V1

Obrázek 6.6: *Zachycení šifrované komunikace*

Také byla ověřena funkčnost revokačních listů, které použijeme v případě, kdy chceme zrušit platnost dříve podepsaného certifikátu, aby nemohl být použit k dalšímu ověřování. Typické důvody zrušení platnosti jsou například: ukončení přístupu uživatele do VPN, soukromý klíč s certifikátem byl odcizen a další.

Revokační list můžeme vygenerovat například pomocí nástroje Easy-RSA. V příkazové řádce se musíme nacházet v adresáři easy-rsa. Pomocí příkazů níže zrušíme platnost certifikátu požadovaného klienta.

```
. ./vars
./revoke-full klient
```

Po zadání příkazů se vygeneruje CRL (Certificate Revocation List) soubor pojmenovaný `crl.pem`. Tento soubor zkopírujeme například do adresáře, kde se nachází konfigurace OpenVPN . V konfiguračním souboru OpenVPN serveru přidáme pro zapnutí CRL ověřování následující řádek.

```
crl-verify crl.pem
```

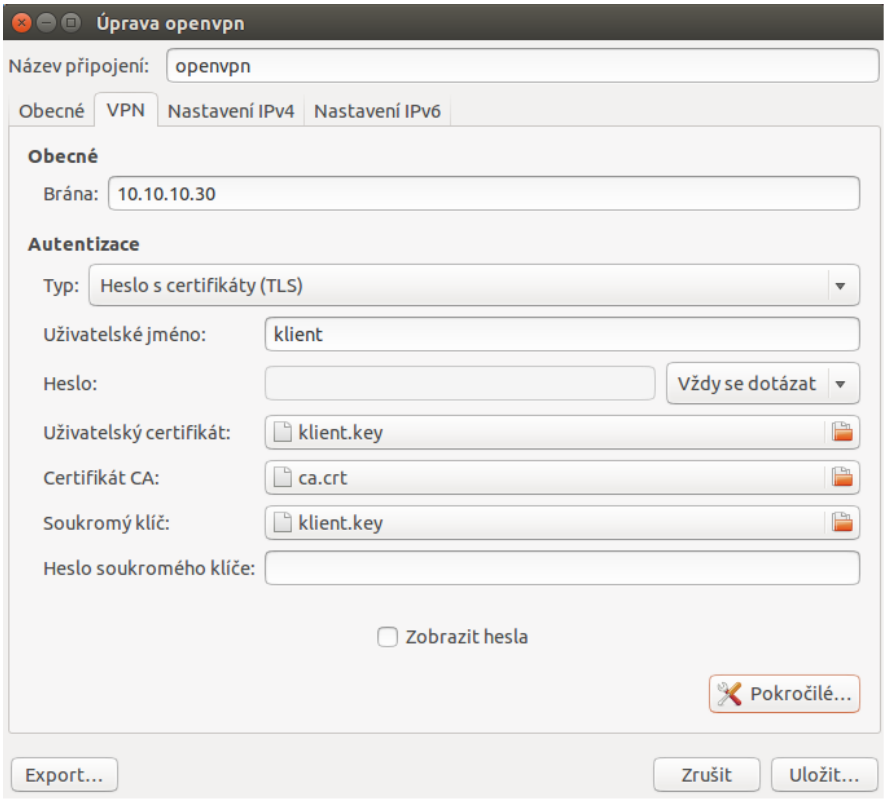
Nyní jsou všichni připojení klienti také ověřováni vůči CRL souboru a jakákoliv pozitivní shoda má za následek odmítnutí spojení. Revokační listy můžeme také generovat pomocí softwaru XCA.

Software OpenVPN nabízí také spoustu dalších nastavení a možností konfigurace, které jsou podrobně popsány v dokumentaci [24] [26]. Tímto byla ověřena funkčnost navrhnutého řešení se softwarem OpenVPN.

Připojování klientů k serveru pomocí uživatelského jména a certifikátů uložených na tokenu bylo ověřeno z operačních systémů Ubuntu 14.04 LTS a Windows 7. Na obou operačních systémech bylo při spuštění OpenVPN a navazování spojení použito prostředí příkazového řádku. Při použití aplikace OpenVPN GUI (verze 5.0) na Windows 7 navázání spojení neproběhlo úspěšně, neboť nás aplikace nedokáže vyzvat k zadání PINU pro přístup na token. V případě lokálně uložených certifikátů na PC aplikace funguje správně.

6.4 Další možnosti a rozšíření

Na operačním systému Ubuntu je také možno doinstalovat grafický doplněk pro OpenVPN. Tento doplněk se poté používá pro vytvoření VPN spojení ve správci sítě. Nastavení lze importovat z konfiguračního souboru nebo vše nastavit ručně. Bohužel doplněk nepodporuje veškeré parametry, které OpenVPN nabízí. Taktéž zde není podpora pro práci s tokenem. Typ autentizace je zde možno nastavit na použití statického klíče, hesla, hesla s certifikáty a samotných certifikátů. Certifikáty musí být uloženy lokálně na PC. Doplněk nainstalujeme pomocí následujícího příkazu `apt-get install network-manager-openvpn` (verze 0.9.8.2). Na obrázku 6.7 je ukázka hlavního okna tohoto doplňku.



Obrázek 6.7: *Ukázka OpenVPN doplňku*

Další řešení, které OpenVPN nabízí, se nazývá OpenVPN Access Server. Jedná se o plně vybavené VPN řešení, které v sobě integruje funkce OpenVPN serveru. Obsahuje webovou správu pro administraci, zjednodušenou aplikaci OpenVPN Connect UI a softwarové balíčky OpenVPN Client. Podporuje připojení klientů z operačních systémů Windows, Mac, Linux, Android a iOS. OpenVPN Access Server můžeme stáhnout pro vyzkoušení z oficiálních stránek [16]. Bohužel při nainstalování je omezen maximální počet pro připojení klientů na dva. V případě že bychom chtěli OpenVPN Acces Server využívat pro více klientů, musíme si zakoupit licenci. Licence pro připojení jednoho klienta na rok stojí 9,60$. Cena zahrnuje také aktualizace a podporu. Je nutný minimální nákup licence pro 10 klientů.

7 Software strongSwan

V následující kapitole je popsán průběh testování softwaru strongSwan (verze 5.1.2), návrh a řešení dvoufaktorové autentizace. Je zde popsána instalace a zprovoznění strongSwan, popis základní konfigurace a konfiguračních souborů. Jak již bylo zmíněno v kapitole 3.2, software strongSwan nabízí autentizaci pomocí sdílených klíčů, digitálních certifikátů nebo uživatelského jména a hesla. Z bližšího pohledu nám tedy strongSwan nabízí tyto způsoby ověřování:

- **Pre-Shared-Key (PSK)** – předsdílený klíč, výhodou je jednoduchá implementace, ale není doporučován pro rozsáhlé nasazení.
- **Public Key Authetication** – využívá RSA nebo ECDSA X.509 certifikáty pro ověření druhého účastníka.
- **Extensible Authentication Protocol (EAP)** – zahrnuje několik možných metod autentizace a jejich pluginy. Některé jsou založeny na autentizaci uživatelského jména a hesla (EAP-MD5, EAP-MSCHAPv2), nebo na autentizaci certifikátů.
- **eXtended Authentication (Xauth) –** poskytuje autentizaci v IKEv1. Používá se především pro autentizaci na základě uživatelského jména a hesla. [36]

Primární zaměření je na využití uživatelského jména a hesla, certifikátů a jejich následné umístění na USB tokenu (použití způsobu Public Key Authetication a EAP metod). Po instalaci a seznámení je popsán postup řešení pro dvoufaktorovou autentizaci. Celý postup konfigurace a návrh řešení je soustředěn na OS Ubuntu (verze 14.04 LTS). V závěru kapitoly je ověřena možnost připojení Windows klientů k strongSwan serveru.

7.1 Instalace strongSwan

V následující podkapitole jsou popsány různé způsoby instalace softwaru strongSwan na operační systém Ubuntu. Jako první a nejednoduší způsob je instalace přes klasickou správu balíku, pomocí následujícího příkazu.

```
apt-get install strongswan
```

Tento příkaz nainstaluje software strongSwan a všechny potřebné balíky pro práci automaticky. Zjistit nainstalovanou je možné pomocí příkazu `ipsec version` nebo `apt-cache policy strongswan`.

Další možností je manuální instalace pomocí kompilace zdrojových kódů. Potřebný soubor stáhneme ze stránek výrobce [19]. Níže se nachází rozbalení archivu pomocí prvního příkazu a přepnutí v terminálu do rozbalené složky.

`tar xzf strongswan-5.1.2.tar.gz` následně `cd strongswan-5.1.2`

Některé programy mohou pro kompilaci vyžadovat další knihovny a instalační balíky. Po rozbalení by se ve složce měl nacházet soubor INSTALL s návodem k instalaci. Mezi

požadovanými balíky jsou kryptografické knihovny `libcrypto`, `libgcrypt` a aritmetická knihovna `libgmp-dev`. V rozbalené složce zadáme pro kompilaci následující příkazy.

```
./configure --prefix=/usr --sysconfdir=/etc --<naše nastavení>
make
make install
```

V prvním příkazu si můžeme zvolit vlastní nastavení, jaké moduly se mají nebo nemají instalovat, například `--enable-eap-md5`, `--disable-des` a další. V tomto případě instalace také závisí na distribuci operačního systému, kde jsou především rozdílné názvy požadovaných balíčků. [37]

StrongSwan podporuje jistou interoperabilitu neboli schopnost spolupracovat s jinými operačními systémy, jako například Windows 7 a novější verze (IKEv2), Windows Vista a starší verze (IKEv1), kdy OS Windows je v roli klienta a lze se připojit k strongSwan serveru.

Počínaje novou verzí, může být strongSwan dokonce sestaven na platformě Windows pomocí MinGW toolchain, což je kompletní runtime prostředí pro GCC. Podporován je Windows 7, Server 2008 R2 a novější vydání. V tomto řešení je značně omezená funkcionalita. Většina modulů a doplňků nebyla importována a zatím testována. Více informací a návod na zprovoznění tohoto řešení se nachází na oficiálních stránkách strongSwan [38].

7.2 Základní konfigurace

V následující podkapitole je popsána základní konfigurace, konfigurační soubory a spuštění služby strongSwan. Po instalaci se nám vytvoří několik adresářů a konfiguračních souborů:

- **ipsec.conf** – soubor poskytuje konfiguraci připojení IPsec. Tento konfigurační soubor se skládá ze tří různých sekcí. První *„config setup“* definuje obecné parametry. Další *„conn <name>“* definuje dané spojení a *„ca <name>“* obsahuje parametry certifikační autority.
- **ipsec.secret** – soubor obsahuje neomezený počet různých druhů zabezpečení. Definuje soukromé klíče RSA, předsdílené klíče PSK, pověření EAP, PIN pro čipové karty, PKCS#12 a další.
- **ipsec.d** – ukládání veškerých certifikátů a soukromých klíčů. Obsahuje podadresáře pro jednotlivé typy souborů, například adresář `private` pro ukládání soukromých klíčů, dále adresář `certs` pro certifikáty a `cacerts` pro certifikáty CA.
- **strongswan.conf** – umožňuje globální nastavení modulů patřičného démona. V tomto souboru definujeme jednotlivé nainstalované moduly, například šifry, hashovací algoritmy, EAP moduly, které potřebujeme pro funkci našeho spojení. U novější verze strongSwan je v tomto souboru defaultně nastaveno načítání všech konfiguračních souboru z adresáře `strongswan.d` a konfigurací jednotlivých modulů z adresáře `strongswan.d/charon`.

- **strongswan.d** – obsahuje konfigurační soubory a důležitý adresář charon, ve kterém se nacházejí konfigurační soubory všech nainstalovaných modulů. Například konfigurační soubor `pkcs11.conf`, `eap-md5` a další.

Veškeré zmíněné konfigurační soubory a adresáře se nacházejí v kořenovém adresáři systému ve složce `/etc`.

V strongSwan je zavedena jistá terminologie. Označení vlevo (left) a vpravo (right) se používá v konfiguračním souboru `ipsec.conf` a označuje dva koncové body IKE SA. Označení „vlevo“ znamená místní peer, na kterém je uložena konfigurace a „vpravo“ je vzdálený peer, ke kterému se připojujeme. Pro snadné zapamatování se můžeme podívat pouze na první písmeno tohoto označení (left = local, right = remote).

Nyní následuje ukázka konfigurace spojení dvou stran (zařízení – zařízení neboli host – host) s využitím autentizace pomocí předsdílenhého klíče (PSK). Na obou počítačích je nainstalován software strongSwan 5.1.2. StrongSwan je defaultně nastaven na tunelovací režim s využitím protokolu ESP. Je možné využít také protokol AH, software strongSwan ale nepodporuje kombinaci obou těchto protokolů AH+ESP. [36] [39]

Konfigurační soubor host-1

Níže se nachází konfigurace souboru `ipsec.conf`. Defaultní parametry pro všechna spojení se vyskytují v *„conn %default“*. Název našeho konkrétního definovaného spojení je „PSK-spojeni“ *v sekci „conn PSK-spojeni“*. V tomto případě není potřeba konfigurovat dříve zmíněné sekce *„config setup“* a *„ca“*.

/etc/ipsec.conf:

```
conn %default
      ikelifetime=60m       #doba pro vypršení IKE SA
      keylife=20m           #doba platnosti klíče
      keyingtries=1         #počet pokusů k vyjednání spojení
      keyexchange=ikev2     #verze protokolu
      compress=yes          #zapnutí komprese
      authby=secret         #autentizace sdíleným klíčem

conn  PSK-spojeni
      left=192.168.0.100    #zdrojová strana (lokální)
      leftfirewall=yes      #povolení pravidel IP tables
      right=192.168.0.200   #cílová strana (vzdálená)
      auto=add              #načtení spojení
```

V souboru `ipsec.conf` může být pouze jedna sekce *„config setup“*, množství sekcí *„conn“* a *„ca“* je neomezené.

Nyní musíme v souboru `ipsec.secrets` definovat náš sdílený tajný klíč. Zadáme naši IP adresu, následně parametr PSK a za ním požadovaný klíč. Klíč je také možné načítat ze souboru zadáním jeho názvu, tento klíč musí být uložen v `ipsec.d/private`. Při reálném používání je vygenerovaný sdílený klíč kvůli bezpečnosti mnohonásobně delší.

/etc/ipsec.secrets

```
192.168.0.100 192.168.0.200 :    PSK 0sFpZAZqEN6Ti9sqt4ZP5EWcqx
```

V souboru `strongswan.conf` probíhá načtení potřebných modulů pro sestavení spojení, nemusíme zde provádět žádné změny. Vidíme, že pro démona charon probíhá načítání konfigurací všech modulů z adresáře `strongswan.d/charon/`.

/etc/strongswan.conf

```
charon {
        load_modular = yes
        pugins {
                include strongswan.d/charon/*.conf
        }
}
include strongswan.d/*.conf
```

Konfigurace druhé strany (host-2) je téměř totožná s konfigurací host-1. Jsou zde pouze rozdílně nastavené IP adresy. Konfigurační soubor `strongswan.conf` a defaultní nastavení z `ipsec.conf` zde již není uvedeno.

Konfigurační soubor host-2

V konfiguračním souboru `ipsec.conf`, jsou rozdílně nastaveny IP adresy. Lokální IP adresa je 192.168.0.200 a vzdálená, která byla nakonfigurována na host-1 je 192.168.0.100.

/etc/ipsec.conf:

```
conn  PSK-spojeni
      left=192.168.0.200    #zdrojová strana (lokální)
      leftfirewall=yes      #povolení pravidel IP tables
      right=192.168.0.100   #cílová strana (vzdálená)
      auto=add              #načtení spojení
```

V souboru `ipsec.secrets` definujeme sdílený klíč. Zadáme naši IP adresu, následně parametr PSK a za ním požadovaný klíč, který je nastaven stejně také na druhé straně spojení.

/etc/ipsec.secrets

```
192.168.0.200 192.168.0.100 :    PSK 0sFpZAZqEN6Ti9sqt4ZP5EWcqx
```

Pomocí parametrů níže nastavíme naše požadované šifrovací a hashovací algoritmy a skupinu Diffie-Hellmanova algoritmu. Prvním parametrem zadáme seznam nastavení pro protokol IKE. V IKEv2 je možnost do tohoto nastavení zahrnout více algoritmů a návrhů. Dalším parametrem zadáme seznam šifrovacích a hashovacích algoritmů pro protkol ESP, které mají být použity pro IPsec spojení. Nastavení se provádí v `ipsec.conf` nejčastěji v části *„conn %default“*.

```
ike=aes256-sha1-modp2048!
esp=aes256-sha1!
```

Co se týče množství zapisování hlášení (úroveň logování), ve výchozím nastavení IKE démon charon provádí logování přes syslog pomocí LOG_AUTHPRIV a LOG_DAEMON. Defaultní úroveň logování pro všechny subsystémy (skupiny) je nastavena na hodnotu 1.

IKE démon zná úrovně logování v rozmezí -1 až 4:

- -1 – absolutně tichý.
- 0 – velmi základní audit logy (např. SA up/down).
- 1 – střední kontrola s chybami, dobrý výchozí vidět, co se děje
- 2 – podrobnější kontrola, ladění.
- 3 – Včetně RAW dat v šestnáctkové soustavě.
- 4 – Obsahuje veškeré data.

Každá zaznamenaná zpráva (log) obsahuje také zdroj, ze kterého subsystému (skupiny) v démonu pochází. Jedná se například o skupiny ike, tls, tnc a další. Konfigurace se provádí v souboru `ipsec.conf` v sekci *„config setup“* použitím parametru „charondebug“.

```
charondebug="ike 2, tls 3, esp 1"
```

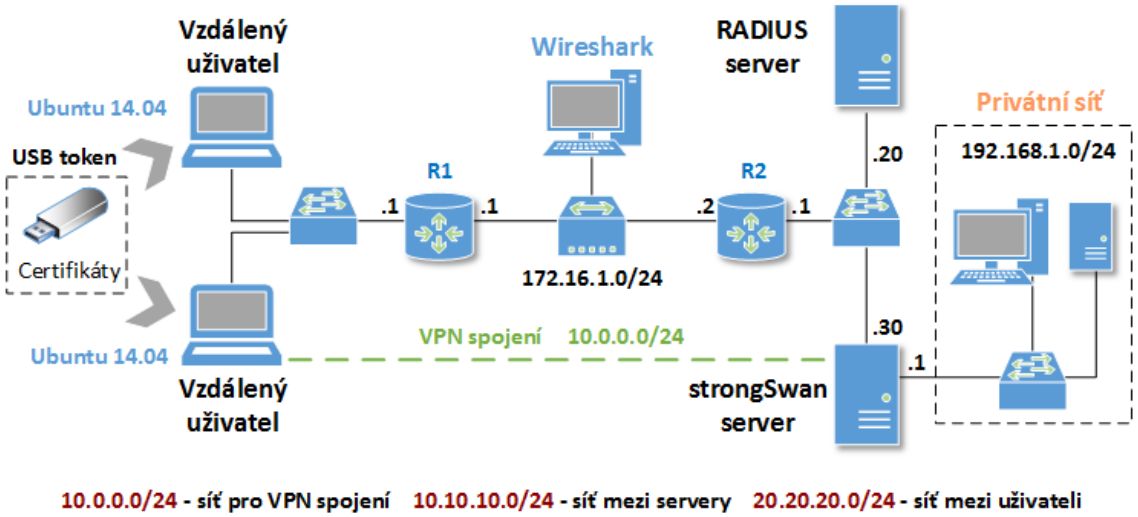
StrongSwan (od verze 4.2.9) poskytuje také mnohem flexibilnější konfiguraci logování v souboru `strongswan.conf`. tato konfigurace má vyšší prioritu než „charondebug“ v `ipsec.conf`. V současné době jsou dva typy jak definovat logování (využití logování do souboru nebo výpis hlášení do syslogu). Ukázka konfigurace logování v souboru `strongswan.conf` se nachází v příloze L, více informací [40].

Pro vypsání nabízených šifrovacích algoritmů a hashovacích funkcí slouží příkaz `ipsec listalgs`, pro vypsání certifikátů a věřených klíčů slouží `ipsec listcerts` a `ipsec listpubkeys`. Při použití příkazu `ipsec listall` dostaneme výpis veškerých již výše zmíněných informací včetně informace o nainstalovaných modulech.

Pokud je konfigurace obou stran hotova, spustíme na obou stanicích IPsec démona pomocí příkazu `ipsec start`. Pokud již démon běží, provedeme restart démona pro provedení našich změn v konfiguraci. Poté příkazem `ipsec up <název spojení>`, vytvoříme zabezpečené spojení (tunel). Pomocí příkazu `ipsec statusall` získáme podrobnější výpis o stavu sestaveného ipsec spojení.

7.3 Návrh řešení - dvoufaktorová autentizace

V následující kapitole je testována dvoufaktorová autentizace pomocí řešení strongSwan na zapojení v laboratorních podmínkách. Je zde popsán kompletní návod na zprovoznění tohoto navrhnutého řešení. Během testování bude použita vlastní CA a certifikáty, stejně jako u řešení OpenVPN. Použitý USB token obsahuje naše klientské certifikáty, které jsme na něj umístili pomocí softwaru XCA. Na všech serverech je použit OS Ubuntu 14.04 (64-bit verze). Na klientských PC je použit OS Ubuntu 14.04 a také je odzkoušeno připojení z OS Windows 7 (64-bit verze). Nakonec je ověřena celková funkčnost tohoto řešení. Na obrázku 7.1 je zobrazeno schéma zapojení.



Obrázek 7.1: *Navrhnuté schéma zapojení*

Na obrázku 7.1 vidíme jednotlivé zařízení a propojení mezi nimi. Nachází se zde strongSwan server, který zpřístupňuje připojeným vzdáleným uživatelům privátní síť. Jedná se tedy o připojení VPN pro vzdálený přístup (zařízení-síť), pomocí strongSwan můžeme také nakonfigurovat další typy spojení, například spojení zařízení – zařízení nebo spojení mezí sítěmi. Klienti jsou autentizováni pomocí digitálních certifikátů a také pomocí uživatelského jména a hesla na RADIUS serveru, při vytváření VPN spojení tedy strongSwan server zasílá požadavky na RADIUS server. Uživatelské jméno a heslo je uloženo v MySQL databázi. U strongSwan je možnost konfigurace několika typů a metod autentizace lokálně nebo externě na RADIUS serveru. V další podkapitole budou popsány možnosti těchto metod a jak jednotlivé typy nakonfigurovat. Na trase mezi serverem a klienty jsou dva směrovače R1 a R2, které obstarávají směrování mezi sítěmi. Dále jsou popsány jednotlivé kroky konfigurace.

7.3.1 Konfigurace strongSwan serveru

Následně je popsán postup konfigurace strongSwan serveru. Předpokládáme, že software strongSwan je nainstalován s defaultním nastavením a požadovanými moduly. Nesmíme také zapomenout uložit certifikát certifikační autority do adresáře `ipsec.d/cacerts`. Jako první provedeme konfiguraci souboru `ipsec.conf`. V sekci *„%default“* nastavíme defaultní parametry, které jsou pro všechna spojení stejná. Určíme dobu, po které vyprší IKE SA (60 minut), dále dobu platnosti klíče na 20 minut a počet pokusů k vyjednání spojení na hodnotu jedna. Pomocí parametru `keyexchange` vybereme způsob výměny klíčů a který protokol by měl být použit pro inicializaci spojení (v tomto případě IKEv2). Následuje nastavení šifrovacího algoritmu, hashovací funkce a parametrů Diffie-Hellman algoritmu pro IKE a pro IPSEC spojení. Spojení bude vytvořeno pouze s těmito nastaveními pro IKE a ESP, klient tedy musí podporovat tyto šifry. Případně můžeme nechat defaultní nastavení a obě strany se dohodnou na nejlepší shodě.

/etc/ipsec.conf

```
conn %default
      ikelifetime=60m
      keylife=20m
      keyingtries=1
      keyexchange=ikev2
      ike=aes256-sha1-modp1024!
      esp=aes256-sha1!
```

Nyní nastavíme naše definované spojení *„tunel“*. První pro naši levou stranu nastavíme fyzickou IP adresu, dále zadáme požadovaný certifikát. Pomocí `leftsubnet` určíme naši privátní síť, kterou bude server nabízet klientům. Parametrem `leftid` nastavíme identifikátor dle certifikátu serveru, poté nastavíme `leftfirewall`. Parametr `leftauth` určí typ autentizace naší levé strany, v tomto případě pomocí *„Public Key Authetication“*, který využívá RSA autentizaci s X.509 certifikáty.

```
conn tunel
      left=10.10.10.30
      leftcert=server.crt
      leftsubnet=192.168.1.0/24
      leftid="C=CZ, ST=NA, L=Ostrava, O=VSB, OU=MOB, CN=server,
      N=DP, E=tab0012@vsb.cz"
      leftfirewall=yes
      leftauth=pubkey
```

Autentizace pro pravou stranu je nastavena na `eap-radius`. StrongSwan server pro autentizaci připojovaných klientů zasílá požadavky na RADIUS server. Jako poslední nastavíme síť 10.0.0.0/24, ze které budou přidělovány virtuální adresy, pomocí parametru `right=%any` určíme připojení jakékoli IP adresy. Nakonec provedeme načtení spojení.

```
rightauth=eap-radius
rightsourceip=10.0.0.0/24
right=%any
auto=add
```

Pokud bychom chtěli provádět autentizaci lokálně, stačí místo `eap-radius` nastavit například `pubkey`, `eap-mschapv2` a další nabízené typy. Je zde také možnost nastavit požadavek na více typů autentizace zároveň následujícím způsobem:

```
rightauth=pubkey
rightauth2=eap-mschapv2
```

Nastavení parametru `rightid` zde nemá smysl, neboť se k serveru připojuje několik klientů a není pevně určen jeden klient. Pomocí `leftsubnet=0.0.0.0/0` můžeme nastavit, aby byl veškerý provoz směrován přes vytvořený VPN tunel.

Nyní musíme v souboru `ipsec.secrets` definovat soukromý klíč serveru `server.key`, klíč musí být uložen v `ipsec.d/private`. V případě lokální autentizace klientů (např. pomocí eap-mshapv2), by řádek `klient : EAP "12345"` sloužil pro ověření klienta, kdy je zadáno jeho jméno a heslo.

/etc/ipsec.secret

```
: RSA server.key

#klient : EAP "12345"
```

Soubor `strongswan.conf`, který načítá veškeré moduly, necháme beze změn. Dále musíme mít nainstalován doplněk `eap-radius` a v konfiguračním souboru provedeme následující úpravy. Jako první přidáme adresu RADIUS serveru a následně nastavíme takzvaný *„secret“*, který musí být nastaven také na RADIUS serveru. Ostatní parametry v konfiguračním souboru necháme v defaultním nastavení. Můžeme zde také nastavit připojení k více RADIUS serverům. [41]

/etc/strongswan.d/charon/eap-radius.conf

```
eap-radius {
      server = 10.10.10.20
      secret = 123456
    }
```

Na serveru nyní nastavíme síťová rozhraní, na kterých zapneme IPv4 směrovaní. Otevřeme konfigurační soubor `sysctl.conf`.

```
gedit /etc/sysctl.conf
```

V tomto konfiguračním souboru odkomentujeme následující řádek.

```
net.ipv4.ip_forward=1
```

Následně provedeme restart síťové služby nebo použijeme příkaz `sysctl -p`. Kompletní konfigurace serveru se nachází v příloze M, také doporučuji dokumentaci s popisem všech parametrů [39]. Dále může být potřeba na serveru nastavit pravidla firewallu např. pomocí iptables, pravidla pro směrování jsou uvedeny v příloze M.

7.3.2 Konfigurace klienta

Nyní je potřeba provést konfiguraci klientské strany. Je zde popsána konfigurace bez sekce *„conn %default“*, která je totožná s konfigurací na serveru. Předpokládáme, že je software strongSwan nainstalován s defaultním nastavením a požadovanými moduly. Na operačním systému Ubuntu 14.04 LTS se vyskytl problém s inicializací USB tokenu, software strongSwan nebyl schopen s tokenem pracovat. Tento problém se vyskytl pouze při instalaci přes správu balíků. Při instalaci pomocí kompilace zdrojových kódů software strongSwan s USB tokenem pracoval správně. Nesmíme také zapomenout uložit certifikát certifikační autority do adresáře `ipsec.d/cacerts`.

Jako první nakonfigurujeme naše definované spojení *„tunel“*. Pro naši levou stranu nastavíme fyzickou IP adresu 20.20.20.10, dále nastavíme požadovaný certifikát. Zde musíme zadat parametr `%smartcard:<keyid>`, který slouží pro práci s tokenem. Klíčové slovo *„keyid“* je ID hodnota certifikátu uloženého na tokenu. Parametrem `leftid` nastavíme identifikátor dle certifikátu klienta nebo lze použít pouze *„SubjectAltName“* certifikátu klient. Pomocí `leftsourceip` určíme získání IP adresy z rozsahu na serveru. Parametr `leftauth` určí typ autentizace naší levé strany (EAP).

/etc/ipsec.conf

```
conn tunel
      left=20.20.20.10
      leftcert=%smartcard:6FF32AB18778BCABBB509E597CCAF3E7817E21B
      #leftid="C=CZ, ST=NA, L=Ostrava, O=VSB, OU=MOB, CN=klient,
      N=DP, E=tab0012@vsb.cz"
      leftid=klient
      leftfirewall=yes
      leftsourceip=%config
      leftauth=eap
```

Autentizace pro pravou stranu je nastavena na `pubkey`. Pomocí parametru `right` nastavíme fyzickou IP adresu serveru. Server musíme být schopni dosáhnout pomocí pingu. Parametrem `rightsubnet` určíme privátní síť, ke které chceme mít přístup. Pomocí `rightid` nastavíme identifikátor dle certifikátu serveru. Jako poslední definujeme AAA identitu použitou při ověřování IKEv2 EAP (jedná se o identitu RADIUS serveru). Tento parametr je nutný, pokud je použito ověření identity serveru (RADIUS server), v případě že není identita shodná s identitou brány (strongSwan server). Nakonec provedeme načtení spojení.

```
rightauth=pubkey
right=10.10.10.30
rightsubnet=192.168.1.0/24
rightid="C=CZ, ST=NA, L=Ostrava, O=VSB, OU=MOB, CN=server,
N=DP, E=tab0012@vsb.cz"
aaa_identity="C=CZ, ST=NA, L=Ostrava, O=VSB, OU=MOB,
CN=radius, N=DP, E=tab0012@vsb.cz"
auto=add
```

V konfiguračním souboru `ipsec.secrets` musíme definovat soukromý klíč klienta. Soukromý klíč je uložen na USB tokenu, který je chráněn PIN kódem. Musíme použít parametr `PIN %smartcard:<keyid>`, pomocí kterého určíme soukromý klíč a zadáme jeho ID hodnotu. Za ID hodnotou následuje PIN, ovšem kvůli bezpečnosti je dobré zadat parametr `%prompt` a k tokenu se budeme muset přihlásit. Pro přihlášení poté slouží příkazy `ipsec rereadsecrets` nebo `ipsec secrets`, jejich použití lze vidět v kapitole 7.3.4 Ověření funkčnosti. Dále je nastaveno uživatelské jméno a heslo. [41]

/etc/ipsec.secret

```
: PIN %smartcard:6FF32AB18778BCABBB509E597CCAF3E7817E21B
"123456" #%prompt
klient : EAP "12345"
```

Soubor `strongswan.conf`, který načítá veškeré moduly, necháme beze změn. Dále musíme mít nainstalovány příslušné EAP moduly a modul `pkcs11`. V souboru `pkcs11.conf` provedeme následující úpravy (přidáme cestu k PKCS#11 modulu). [42] Kompletní konfigurace klienta se nachází v příloze M.

/etc/strongswan.d/charon/pkcs11.conf

```
modules {
      pkiclient {
            path = /usr/lib64/libeTPkcs11.so
      }
}
```

7.3.3 Konfigurace FreeRADIUS

Nyní provedeme základní konfiguraci RADIUS serveru. RADIUS server bude opět využívat MySQL databázi pro uložení uživatelských jmen a hesel. Veškerý postup a instalace již byla dříve probrána v kapitole 6.3.3. Jako první nastavíme klienta (strongSwan server) v konfiguračním souboru `clients.conf`. Nastavíme IP adresu strongSwan serveru, dále takzvaný *„secret“* pro bezpečnost a nakonec nastavíme jméno.

/etc/freeradius/clients.conf

```
client 10.10.10.30 {
  secret    = 123456
  shortname = strongSwanServer
}
```

Dále musíme mít v souboru `radiusd.conf` odkomentováno načítání konfiguračního souboru `eap.conf`.

/etc/freeradius/radiusd.conf

```
$INCLUDE eap.conf
```

V konfiguračním souboru `eap.conf` nastavíme metody autentizace. Využijeme EAP-PEAP s autentizací pomocí uživatelského jména a hesla (MSCHAPv2 nebo možnost MD5) a klientského certifikátu. V sekci `eap {}` nastavíme defaultní typ na PEAP. Dále v podsekci `tls {}` musíme zadat certifikát CA, soukromý klíč RADIUS serveru a také jeho certifikát, které jsme si vygenerovali. Soubory se nachází v adresáři `/etc/freeradius/certs`.

/etc/freeradius/eap.conf

```
eap {
  default_eap_type = peap
  tls {
    certdir = ${confdir}/certs
    cadir = ${confdir}/certs
    private_key_file = ${certdir}/radius.key
    certificate_file = ${certdir}/radius.crt
    CA_file = ${cadir}/ca.crt
    cipher_list = "DEFAULT"
    dh_file = ${certdir}/dh
  }
```

Nyní v podsekci `peap {}` nastavíme základní typ autentizace na MSCHAPv2, ostatní parametry ponecháme v defaultním nastavení.

```
  peap {
    default_eap_type = mschapv2
    use_tunneled_reply = yes
    virtual_server = "inner-tunnel"
  }
}
```

Nakonec musíme v souboru `default` v sekci `authorize {}` přidat konfiguraci, pomocí které zajistíme, že při autentizaci bude vyžadován také certifikát klienta. [41]

/etc/freeradius/sites-enabled/default

```
update control {
     EAP-TLS-Require-Client-Cert = Yes
}
```

7.3.4 Ověření funkčnosti

V následující podkapitole je ověřena funkčnost nakonfigurovaného řešení. Pokud je všechna konfigurace hotova, spustíme na obou stranách IPsec démona pomocí příkazu `ipsec start`. Pokud již démon běží, provedeme restart démona pro provedení našich změn v konfiguraci. Pomocí příkazu `ipsec restart –nofork` můžeme vidět spouštění démona charon, načítání certifikátů, modulů a další nastavení, výpis příkazu v příloze N.

Poté příkazem `ipsec up <název spojení>` z klientovy strany vytvoříme zabezpečené spojení (tunel). Pomocí příkazu `ipsec status` nebo `ipsec statusall` získáme podrobnější výpis o stavu sestaveného spojení. Bez aktivního spojení obsahuje `ipsec statusall` základní přehled o modulech a nastavení, výpis příkazu ze serveru se nachází v příloze N. Následně je popsán postup navázání spojení z klientské strany. Před zahájením spojení je dobré ověřit, zda byl USB token rozpoznán, výpis z logu `restart --nofork` by měl obsahovat následující informace, kompletní výpis v příloze N.

```
loaded PKCS#11 v2.1 library 'pkiclient'
(/usr/lib64/libeTPkcs11.so)
 Aladdin Ltd.: eToken PKCS#11 v5.0
 found token in slot 'pkiclient':0 (Aladdin eToken PRO 32 00 00)
     Aladdin (Aladdin Ltd.: eToken)
     loaded trusted cert 'klient'
```

Nyní je potřeba se k USB tokenu přihlásit ještě před navázáním spojení. Strongswan se během navazování spojení není schopen dotázat na heslo k tokenu. Přihlásíme se pomocí příkazu `ispec secret` nebo `ipsec rereadsecrets`, níže lze vidět přihlášení k tokenu.

```
Login to '%smartcard:6FF32AB18778BCABBB509E597CCAF3E7817E201B'
required
PIN:
```

Poté příkazem `ipsec up tunel` vytvoříme zabezpečený tunel. Následně je možné vidět pouze některé zprávy z obsáhlého logu při připojení klienta, kompletní výpisy logu ze strany klienta se nachází v příloze N. Jako první se provede zahájení IKE_SA na IP adresu serveru, poté se na server vygeneruje požadavek IKE_SA_INIT.

```
initiating IKE_SA tunel[1] to 10.10.10.30
generating IKE_SA_INIT request 0 [ SA KE No N(NATD_S_IP)
```

Dále jsou obdrženy a zaslány požadavky na certifikáty. V logu jsou také vidět zdrojové a cílové adresy (použité porty) vzájemné komunikace.

```
sending packet: from 20.20.20.10[4500] to 10.10.10.30[4500]
received packet: from 10.10.10.30[4500] to 20.20.20.10[4500]
```

Server je ověřován pomocí autentizace *„Public Key Authentication“*, po obdržení certifikátu serveru je certifikát ověřen.

```
authentication of 'C=CZ, ST=NA, L=Ostrava, O=VSB, OU=MOB,
CN=server, N=DP, E=tab0012@vsb.cz' with RSA signature successful
```

Během komunikace se vytváří několik požadavků IKE_SA_INIT, server také zašle požadavky konkrétní autentizace pro klienta (PEAP, MSCHAPV2).

```
server requested EAP_PEAP authentication (id 0x01)
server requested EAP_MSCHAPV2 authentication (id 0x08)
```

Na konci výpisu obdržíme zprávu o sestavení tunelu mezi klientem a strongSwan serverem, dále dobu životnosti IKE_SA, přidělení virtuální adresy a jako poslední zprávu „*established successfully“*, tedy že spojení bylo úspěšně navázáno.

```
IKE_SA tunel[1] established between
20.20.20.10[klient]...10.10.10.30[C=CZ, ST=NA, L=Ostrava, O=VSB,
OU=MOB, CN=10.10.10.30, N=DP, E=tab0012@vsb.cz]
maximum IKE_SA lifetime 3421s
installing new virtual IP 10.0.0.1
connection 'tunel' established successfully
```

Příkazem `ipsec statusall` získáme výpis o stavu ipsec spojení. Ve výpisu obdržíme název našeho spojení *„tunel“* a infromaci že naslouchámе na IP adrese 20.20.20.10. V sekci *„Connections“* vidíme použití protokolu IKEv2, pro naši levou stranu (lokální) použití EAP autentizace a údaje o použitém certifikátu. Pro pravou stranu (vzdálenou) je použita autentizace *„Public Key Authentication“*.

```
Listening IP addresses: 20.20.20.10
Connections:
       tunel:  20.20.20.10...10.10.10.30  IKEv2
       tunel:  local:  [klient] uses EAP authentication
       tunel:  cert:  "C=CZ....O=VSB, OU=MOB, CN=klient, N=DP"
       tunel:  remote: [server] uses public key authentication
       tunel:  child:  dynamic === 192.168.1.0/24 TUNNEL
```

V sekci *„Security Associations“* je naše aktivní navázané ipsec spojení. Nachází se zde opět údaje o úspěšném navázání spojení mezi klientem a serverem, dále použité šifrovací algoritmy a hashovasí funkce. Vidíme také, že se jedná o tunelovací režim s ESP protokolem. Poslední řádek udává navázání spojení mezi naší přidělenou virtuální IP adresou 10.0.0.1 a privátní sítí 192.168.1.0/24. Kompletní výpis ze strany klienta a obdobný výpis ze strany strongSwan serveru lze nalézt v příloze N.

```
Security Associations (1 up, 0 connecting):
       tunel[2]: ESTABLISHED 3 minutes ago,
20.20.20.10[klient]...10.10.10.30[C=CZ...O=VSB, CN=server, N=DP]
       tunel[2]: IKEv2 SPIs: 8a62ea8db0190294_i*
9dbc3bb7d4360b87_r, EAP reauthentication in 44 minutes
       tunel[2]: IKE proposal:
AES_CBC_256/HMAC_SHA1_96/PRF_HMAC_SHA1/MODP_1024
       tunel{2}:  INSTALLED, TUNNEL, ESP SPIs: c5db408a_i
       tunel{2}:  AES_CBC_256/HMAC_SHA1_96, 0 bytes_i, 0
bytes_o, rekeying in 3 minutes
       tunel{2}:   10.0.0.1/32 === 192.168.1.0/24
```

Nyní se podíváme na zachycenou komunikaci pomocí software Wireshark. Na obrázku 7.2 je zobrazeno několik zpráv při navazování spojení mezi klientem a serverem. Jsou zde IP adresy klienta (20.20.20.10) a serveru (10.10.10.30). Jsou zde pakety ISAKMP protokolu a v něm zapouzdřeny dva druhy zpráv IKE_SA_INIT a IKE_AUTH. První dvě zprávy IKE_SA_INIT z obrázku 7.2 slouží pro vyjednání IKE SA spojení. Tato zpráva od klienta obsahuje například zaslané podporované šifrovací algoritmy, hashovací funkce a skupinu Diffe-Helmanova algoritmu. V našem případě byly tyto parametry pevně nastaveny v konfiguračních souborech na `aes256-sha1-modp1024`.

Následně je vytvořeno zabezpečené spojení IKE SA, zprávy IKE_AUTH jsou již šifrované. Pomocí těchto zpráv se již vyjednává IPsec SA (child sa). Obsah zprávy IKE_SA_INIT je uveden v příloze N.

Source	Destination	Protocol	Length	Info
20.20.20.10	10.10.10.30	ISAKMP	346	IKE_SA_INIT MID=00 Initiator Request
10.10.10.30	20.20.20.10	ISAKMP	379	IKE_SA_INIT MID=00 Responder Response
20.20.20.10	10.10.10.30	ISAKMP	458	IKE_AUTH MID=01 Initiator Request
10.10.10.30	20.20.20.10	ISAKMP	338	IKE_AUTH MID=01 Responder Response
20.20.20.10	10.10.10.30	ISAKMP	282	IKE_AUTH MID=02 Initiator Request
10.10.10.30	20.20.20.10	ISAKMP	1146	IKE_AUTH MID=02 Responder Response

Obrázek 7.2: *Zachycené zprávy při vytváření VPN spojení*

Na obrázku 7.3 je zobrazena zachycená komunikace mezi strongSwan a RADIUS serverem. Jedná se o komunikaci, která nastala během připojování klientské stanice s uživatelským jménem „klient“, kdy OpenVPN server pošle požadavek na ověření RADIUS serveru.

Source	Destination	Protocol	Length	Info
10.10.10.30	10.10.10.20	RADIUS	349	Access-Request(1) (id=44, l=307)
10.10.10.20	10.10.10.30	RADIUS	1132	Access-Challenge(11) (id=44, l=1090)
10.10.10.30	10.10.10.20	RADIUS	195	Access-Request(1) (id=45, l=153)
10.10.10.20	10.10.10.30	RADIUS	1128	Access-Challenge(11) (id=45, l=1086)
10.10.10.30	10.10.10.20	RADIUS	248	Access-Request(1) (id=53, l=206)
10.10.10.20	10.10.10.30	RADIUS	210	Access-Accept(2) (id=53, l=168)

Obrázek 7.3: *Zachycená komunikace s RADIUS serverem*

Na obrázku 7.4 je zachycena šifrovaná komunikace, jedná se o komunikaci mezi klientem a privátní sítí 192.168.1.0. V tomto případě nelze vidět, o jaký typ komunikace se jedná. Vidíme pouze IP adresu klienta, IP adresu strongSwan serveru a protkol ESP, pomocí kterého je komunikace zabezpečena. Za serverem do privátní sítě je komunikace již nešifrovaná.

Source	Destination	Protocol	Length	Info
20.20.20.10	10.10.10.30	ESP	166	ESP (SPI=0xe36de4de)
10.10.10.30	20.20.20.10	ESP	166	ESP (SPI=0xe36de4de)
20.20.20.10	10.10.10.30	ESP	166	ESP (SPI=0xe36de4de)
10.10.10.30	20.20.20.10	ESP	166	ESP (SPI=0xe36de4de)

Obrázek 7.4: *Zachycení šifrované komunikace*

Tímto byla popsána konfigurace a ověření funkčnosti na operačním systému Ubuntu. Dále byla také ověřena funkčnost připojení klientů z operačního systému Windows 7 ke správně nakonfigurovanému strongSwan serveru. Operační systém Windows 7 plně podporuje protokol IKEv2. Starší systém Windows Vista podporuje protokol IKEv1, pro starší systémy Windows se doporučuje použití softwaru třetích stran (např. ShrewSoft VPN Client). Nové připojení k strongSwan serveru se poté vytvoří přes *„Centrum síťových připojení“* ve Windows.

Co se týče vytvoření připojení z operačních systémů Windows, jsou zde určité požadavky a omezení, které budou následně popsány. Jako první jsou zde určité požadavky na certifikát serveru (VPN brány). Vygenerovaný certifikát musí obsahovat u rozšířeného využití

klíče (Extended key usage – EKU) parametr *„TLS Web Server Authentication“*. Tato volba rozšířeného využití klíče je pro správné navázání VPN spojení nutná. Další požadavek na certifikát serveru je, aby *„subjectDistinguishedName“* nebo *„subjectAltName“* obsahoval IP adresu serveru, případně jeho hostname. Pokud tyto parametry nejsou správně nastaveny, spojení není navázáno a klient obdrží zprávu s chybou 13801. [43]

Připojení z operačního systému Windows 7 může být provedeno s následujícími autentizačními metodami:

- **Public Key Authentication** – využívá RSA, označováno jako X.509 certifikáty počítače. V systému Windows vybereme ověřování pomocí možnosti – použít certifikáty počítače.
- **EAP-TLS –** autentizace pomocí EAP-TLS, označováno jako autentizace využívající X.509 uživatelské certifikáty. V systému Windows vybereme ověřování pomocí protokolu EAP a následně možnost – čipová karta nebo jiný certifikát, poté ve vlastnostech nastavíme, že chceme použít čipovou kartu (USB token).
- **EAP-MSCHAP v2** – při této autentizaci se využije uživatelské jméno a heslo. V systému Windows vybereme protokol EAP a následně EAP-MSCHAP v2.

Při připojení z operačního systému Windows 7 lze využít vždy jeden typ autentizace zmíněný výše. Nelze zde použít autentizaci EAP-MSCHAP v2 a zároveň také ověření pomocí certifikátů jako v případě konfigurace na OS Ubuntu. Také zde nejde využít připojení pomocí protokolu PEAP. Jsou zde také nároky na parametry konfiguračního souboru strongSwan serveru, ukázka konfigurace serveru pro EAP-TLS je v příloze O. Kompletní návody pro připojení z OS Windows jsou uvedeny na oficiálních stránkách strongSwan [44].

Software strongSwan nabízí spoustu dalších nastavení a možností konfigurace, které jsou podrobně popsány v dokumentaci [39] [41]. Tímto byla ověřena funkčnost navrhnutého řešení se softwarem strongSwan. Připojování klientů k serveru pomocí uživatelského jména a certifikátů uložených na tokenu bylo ověřeno z OS Ubuntu 14.04 LTS. Pro navázání spojení bylo použito prostředí příkazového řádku. Z OS Windows 7 byly otestovány již výše zmíněné typy autentizací. Při použití USB tokenu bylo spojení úspěšně navázáno a OS Windows 7 nás bez problému vyzve k vložení tokenu do USB portu a následné zadání hesla. V případě jednoho serveru a požadavku připojování klientů jak z OS Ubuntu, tak z OS Windows s rozdílními typy autentizace, musíme nakonfigurovat v souboru `ipsec.conf` více sekcí (spojení) *„conn“*.

7.4 Další možnosti a rozšíření

7.4.1 Grafický doplněk

Na operačním systému Ubuntu je také možno doinstalovat grafický doplněk pro software strongSwan. Tento doplněk se poté používá pro vytvoření VPN spojení ve správci sítě. Nastavuje se zde pouze několik základních parametrů (IP adresa serveru, certifikát CA, typ autentizace a další). Typ autentizace je možno nastavit na použití předsdíleného klíče, hesla a uživatelského jména pomocí EAP, certifikátu a privátního klíče, ale také použití USB tokenu

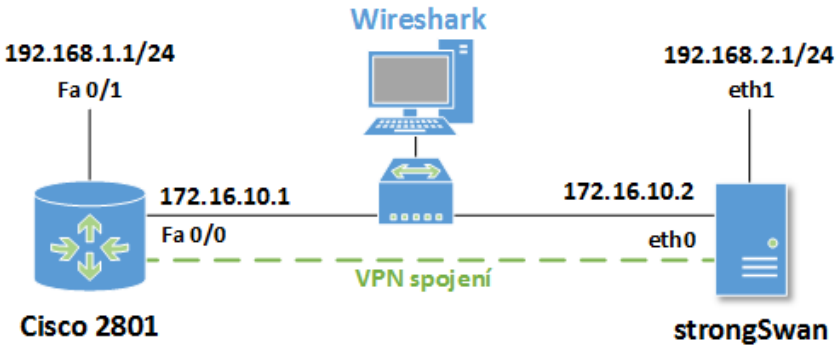
(SmartCard). Typ autentizace jde vybrat vždy jeden, nelze mezi sebou kombinovat více typů zároveň. Doplněk nainstalujeme pomocí následujícího příkazu `apt-get install network-manager-strongswan` (verze 1.3.0). Po grafické stránce je doplněk podobný doplňku OpenVPN z kapitoly 6.4. Pro správnou funkčnost tohoto grafického doplňku musíme přidat do konfiguračního souboru `/etc/strongswan.conf` následujícího démona, tento doplněk nevyužívá klasického démona charon, ale speciálního démona charon-nm.

```
charon-nm {
        load_modular = yes
        pugins {
                include strongswan.d/charon/*.conf
        }
}
include strongswan.d/*.conf
```

Dále je nutné splnit několik požadavků pro správnou funkci USB tokenu. Token obsahuje dvojici klíčů (privátní a veřejný). Oba tyto klíče musí mít nastavené ID na hodnotu, kterou obsahuje certifikát ve svém *„subjectKeyIdentifier“*. Veřejný klíč musí být čitelný bez přihlašování (loginu) na token. Certifikát klienta musí také obsahovat u rozšířeného využití klíče (Extended key usage – EKU) parametr *„TLS Web Client Authentication“*. [45]

7.4.2 Kompatibilita s jiným řešením IPsec

Jak již bylo zmíněno, software strongSwan spolupracuje s jinými implemetacemi IPsec. V laboratorních podmínkách byla otestována kompatibilita se směrovačem Cisco 2801. Na obrázku 7.5 je zobrazeno zapojení při testování kompatibility. Mezi strongSwan serverem a směrovačem Cisco bylo úspěšně navázáno zabezpečené IPsec spojení pomocí sdíleného klíče.



Obrázek 7.5: *Schéma zapojení pro ověření kompatibility*

Při konfiguraci byl použit prokol IKEv1, dostupná verzce Cisco IOS na směrovači nepodporovala protokol IKEv2. Protokol IKEv2 lze nakonfigurovat například na Cisco IOS 15.3T. [46] Veškeré konfigurace a výpisy z navázaného spojení jsou v příloze P.

8 Srovnání OpenVPN a strongSwan

Řešení OpenVPN a strongSwan jsou již na první pohled rozdílné v protokolech, které využívají pro svou funkci. Software OpenVPN používá protokol SSL/TLS a software strongSwan využívá protkol IPsec. Obě tyto řešení jsou open source. Co se týče podpory operačních systémů, software OpenVPN nabízí větší podporu než strongSwan, jedná se především o OS Windows. V případě OpenVPN musíme brát ohled na to, že aplikace musí být nainstalovaná na všech zařízeních (server, klientské PC, mobilní telefon atd.). K strongSwan serveru jsme schopni se z klientského zařízení připojit pouze pomocí správce sítě (kompatibilita s jiným řešením IPsec). Syntaxe a terminologie konfigurace pomocí softwaru OpenVPN a strongSwan jsou rozdílné. Přehled srovnání se nachází v tabulce 8.1.

Tabulka 8.1: *Srovnání open source softwaru OpenVPN a strongSwan*

	Software OpenVPN	**Software strongSwan**
Použitá technologie	SSL/TLS	IPsec
Nabízené typy autentizací	Sdílený klíč, certifikáty, jméno/heslo	Sdílený klíč, certifikáty, jméno/heslo
Podpora tokenu v příkazovém řádku	Ano	Ano
Automatická interaktivita k tokenu v příkazovém řádku	Ano	Ne
Grafické doplňky	Ano	Ano
Ověření zároveň pomocí více typů autentizace v grafickém doplňku	Ne	Ne
Podpora tokenu v grafickém doplňku	Ne	Ano
Propojení se serverem RADIUS	Ano	Ano
Kompatibilita s jiným VPN řešením	Ne	Ano

Pro práci s USB tokenem u OpenVPN není potřeba instalovat další rozšíření, u softwaru strongSwan je potřeba doinstalovat doplněk PKCS#11. V příkazovém řádku (OS Ubuntu) při vytváření spojení OpenVPN automaticky vypíše hlášení pro vložení tokenu nebo zapsání přístupového hesla na token, v případě softwaru strongSwan se musíme nejprve samostatně k tokenu přihlásit a poté vytvořit spojení. Obě řešení podporují na OS Ubuntu instalaci grafického doplňku ve správci sítě. Jde zde zvolit pouze jedna metoda autentizace, nelze je mezi sebou kombinovat. Nevýhodou u OpenVPN je, že v grafickém doplňku není zavedena podpora USB tokenu. Na OS Windows 7 grafická aplikace pro OpenVPN opět nepodporuje token, připojení s využitím tokenu z OS Windows 7 pomocí správce sítě k serveru strongSwan proběhlo bez problému. V navrhnutém řešení byl také u těchto softwarů úspěšně implementován server RADIUS. Podrobněji jsou tyto jednotlivé problémy popsány v průběhu konfigurace a ověřování těchto softwarů.

Závěr

V rámci své diplomové práce jsem v jednotlivých kapitolách splnil cíle, které byly určeny. V úvodní kapitole jsem se zaměřil na popis virtuálních privátních sítí. Popsal jsem zde kladené požadavky na virtuální privátní sítě, jejich výhody a důvody nasazení. Po seznámení s virtuálními privátními sítěmi je popsána vícefaktorová autentizace a její základní metody ověřování (znalost, vlastnictví, biometrika). Po celkovém přehledu virtuálních privátních sítí a vícefaktorové autentizace jsem začal pracovat na hlavním úkolu této práce a to praktickém návrhu a realizaci dvoufaktorové autentizace ve VPN sítích.

V praktické části jsem se jako první věnoval generování potřebných certifikátů a vytvoření vlastní CA pro testování. Použit pro to byl nástroj Easy RSA a software XCA. Nástroj Easy RSA byl dříve součástí instalace OpenVPN, od novější verze je ale potřeba provést instalaci samostatně. Společne se strongSwan je také nainstalován nástroj ipsec pki pro generování a práci s certifikáty. Dále jsem se věnoval zprovoznění USB tokenu na operačním systému Ubuntu a Windows. Na OS Ubuntu byl popsán postup zprovoznění se sadou knihoven OpenSC, s originálními ovladači tokenu, ale také ovladači OpenCT. OpenSC a OpenCT podporuje velké množství tokenů a čipových karet, postup práce je tedy univerzální a lze jej použít pro více zařízení. Na OS Windows 7 byly nainstalovány originální ovladače a token byl operačním systémem bez problému inicializován.

Poté již probíhala práce s řešením OpenVPN a strongSwan. Na vylepšování softwarů se vedle hlavních vývojářů podílí také komunita. Software OpenVPN nabízí možnost webové správy serveru, která je ovšem placená. Co se týče konfigurace softwarů, dle mého vlastního názoru je postup a pochopení konfigurace OpenVPN jednodušší. U řešení OpenVPN a strongSwan byla ověřena funkčnost použití USB tokenu (dvoufaktorové autentizace) a jejich možnosti. Jsou zde na token kladeny určité nároky a požadavky pro správnou funkcionalitu. Obě řešení mají své vlastní výhody a nevýhody. Budoucí uživatel (správce) zvolí řešení dle jeho požadavků, vhodnosti nasazení, použitých operačních systémů a intuitivnosti ovládání. Výhodou u softwaru OpenVPN je lepší interaktivita vůči USB tokenu v příkazovém řádku. OpenVPN je jednotné řešení, musí být nainstalováno na všech stanicích. Velkou nevýhodu vidím v tom, že v grafickém doplňku není implementována podpora USB tokenů. Výhodou softwaru strongSwan je kompatibilita s jiným IPsec řešením a podpora USB tokenu v grafickém doplňku, v příkazovém řádku je vůči uživateli přívětivost horší.

Přínosem této diplomové práce pro mne byla možnost vyzkoušet dvoufaktorovou autentizaci s USB tokenem. Díky práci jsem se blíže seznámil s konfigurací a možnostmi softwaru OpenVPN a strongSwan. Tuto zkušenost beru jako praxi do budoucna v oblasti počítačových sítí. Přínos této diplomové práce v odvětví virtuálních privátních sítí a autentizace vidím v tom, že byla ověřena funkčnost dvoufaktorové autentizace s použitým open source řešením. Pro toho, kdo chce využít open source řešení VPN sítí a dvoufaktorovou autentizaci, může do začátku tato práce sloužit jako přehled a návod na konfiguraci využitých softwarů.

Použitá literatura

[1] TECHNET MICROSOFT. Virtuální privátní sítě: Co je VPN?. Knihovna Technet Microsoft [online]. [cit. 2014-09-25]. Dostupné z: http://technet.microsoft.com/cs-cz/library/cc731954(v=ws.10).aspx

[2] NORTHCUTT, Stephen, Lenny ZELTSER, Scott WINTERS, Koren KENT FREDERICK a Ronold W. RITCHEY. Bezpečnost počítačových sítí. Brno: CP Books, 2005. ISBN 80-251-0697-7.

[3] LUHOVÝ, Karel. Tutoriál - Virtuální privátní sítě VPN. Svět sítí [online]. [cit. 2014-09-25]. Dostupné z: http://www.svetsiti.cz/clanek.asp?cid=VPN-1-historie-definice-a-duvody-budovani-612003

[4] WEB4COMPANY. Bezpečnost: Autentizace a autorizace. Web4Company [online]. [cit. 2014-09-25]. Dostupné z: http://www.web4company.cz/bezpecnost-autentizace-autorizace/

[5] IBM. Tivoli Provisioning Manager 5.1.1 - Zabezpečení: Zabezpečení uživatele - Co je autorizace?. IBM - Infocenter [online]. [cit. 2014-09-25]. Dostupné z: http://publib.boulder.ibm.com/infocenter/tivihelp/v20r1/index.jsp?topic=%2Fcom.ibm.tivoli.tpm.sec.doc%2Fusers%2Fcsec_userauth.html&lang=cs

[6] IBM. Tivoli Provisioning Manager 5.1.1 - Zabezpečení: Zabezpečení uživatele - Dvoufaktorová autentizace. IBM - Infocenter [online]. [cit. 2014-09-25]. Dostupné z: http://publib.boulder.ibm.com/infocenter/tivihelp/v20r1/index.jsp?topic=%2Fcom.ibm.tivoli.tpm.sec.doc%2Fusers%2Fcsec_twofactor.html

[7] WEB4COMPANY. Bezpečnost: Šifrování. Web4Company [online]. [cit. 2014-09-25]. Dostupné z: http://www.web4company.cz/bezpecnost-sifrovani/

[8] ŘEZÁČ, Filip, Miroslav VOZŇÁK a Jan ROZHON. Skripta Bezpečnost v komunikacích [online]. Ostrava, 2013 [cit. 2014-10-09]. ISBN ISBN skript.

[9] TECHNET MICROSOFT. Zabezpečení protokolu IP: Integrita dat pomocí funkcí hash. Knihovna Technet Microsoft [online]. [cit. 2014-10-09]. Dostupné z: http://technet.microsoft.com/cs-cz/library/cc736330(v=ws.10).aspx

[10] PRŮCHA, Ondřej. Požadavky kladené na VPN. Vše kolem VPN [online]. 2005 [cit. 2014-10-09]. Dostupné z: http://home.zcu.cz/~ondrous/index.php?vyber=0

[11] MACHNÍK, Petr. Širokopásmové sítě pro integrovanou výuku VUT a VŠB-TUO [online]. 1. vyd. Ostrava, 2014 [cit. 2014-10-09]. ISBN 978-80-248-3630-0. Dostupné z: http://comtech.vsb.cz/moodle/pluginfile.php/2361/mod_resource/content/1/%C5%A0irokop%C3%A1smov%C3%A9%20s%C3%ADt%C4%9B.pdf

[12] TABAČEK, Michal. Srovnání parametrů a funkcí směrovačů Huawei a Cisco. 2013. Dostupné z: http://dspace.vsb.cz/. Bakalářská práce. VŠB – Technická univerzita Ostrava.

[13] SYME, Matthew a Philip GOLDIE. Optimizing network performance with content switching: server, firewall, and cache load balancing. Upper Saddle River, NJ: Prentice Hall, 2004, xxii, 262 p. ISBN 01-310-1468-4.

[14] DIERKS, T. a E. RESCORLA. The Transport Layer Security (TLS) Protocol Version 1.2. *RFC* [online]. 2008 [cit. 2014-11-23]. Dostupné z: http://tools.ietf.org/html/rfc5246

[15] OPENMANIAK. Co je OpenVPN?: SSL. Tutoriál OpenVPN [online]. 2008 [cit. 2014-11-23]. Dostupné z: http://openmaniak.com/cz/openvpn.php

[16] OpenVPN Community Software. OPENVPN TECHNOLOGIES, Inc. OpenVPN. *OpenVPN* [online]. © 2002-2015 [cit. 2014-11-23]. Dostupné z: https://openvpn.net/index.php/open-source.html

[17] HLADÍK, Radek. OpenVPN - VPN jednoduše - Root.cz. Root.cz - informace nejen ze světa Linuxu [online]. 2004 [cit. 2014-11-23]. Dostupné z: http://www.root.cz/clanky/openvpn-vpn-jednoduse/#ic=articles-related&icc=item-1

[18] STRONGSWAN. StrongSwan - About.[online]. 2015 [cit. 2014-11-23]. Dostupné z: https://www.strongswan.org/about.html

[19] STRONGSWAN. The OpenSource IPsec-based VPN Solution. StrongSwan - IPsec for Linux [online]. 2014 [cit. 2014-11-23]. Dostupné z: https://www.strongswan.org/

[20] BAY, Robin. Čipové karty a USB tokeny, aneb bezpečnější autentizace a šifrování. Svět sítí: informace ze světa počítačových sítí [online]. 2003 [cit. 2015-01-06]. Dostupné z: http://www.svetsiti.cz/clanek.asp?cid=Cipove-karty-a-USB-tokeny-aneb-bezpecnejsi-autentizace-a-sifrovani-1-ucel-cipovych-karet-a-USB-tokenu-3072003

[21] Obrázek - Aladdin eToken PRO 32K. Https://secdiary.com/ [online]. [cit. 2015-01-06]. Dostupné z: https://secdiary.com/wp-content/uploads/2012/01/EToken_PRO_USB.jpg

[22] EToken PRO - User Access Control Token. SAFENET, Inc. SafeNet: Certificate-Based PKI USB Authenticators [online]. [cit. 2015-01-06]. Dostupné z: http://www.safenet-inc.com/multi-factor-authentication/authenticators/pki-usb-authentication/etoken-pro/#

[23] Downloads: OpenVPN 2.3.6. OPENVPN TECHNOLOGIES, Inc. OpenVPN [online]. [cit. 2015-01-06]. Dostupné z: http://openvpn.net/index.php/open-source/downloads.html

[24] Documentation. OPENVPN TECHNOLOGIES, Inc. OpenVPN [online]. [cit. 2015-01-06]. Dostupné z: http://openvpn.net/index.php/open-source/documentation.html

[25] BUDIŠ, Petr. Elektronický podpis a jeho aplikace v praxi. 1. vyd. Olomouc: ANAG, 2008, 157 s. ISBN 978-80-7263-465-1.

[26] Documentation - HOWTO. *OpenVPN* [online]. © 2002-2013 [cit. 2015-01-24]. Dostupné z: https://openvpn.net/index.php/open-source/documentation/howto.html

[27] User Documentation - ipsec pki. *StrongSwan* [online]. © 2006-2014 [cit. 2015-01-24]. Dostupné z: https://wiki.strongswan.org/projects/strongswan/wiki/IpsecPKI

[28] HOHNSTÄDT, Christian. XCA - X Certificate and key management. *XCA* [online]. 2014 [cit. 2015-03-11]. Dostupné z: http://xca.sourceforge.net/

[29] OpenSC – tools and libraries for smart cards. *OpenSC* [online]. 2014 [cit. 2015-03-11]. Dostupné z: https://github.com/OpenSC/OpenSC/wiki

[30] Quick Start with OpenSC. *OpenSC* [online]. 2011 [cit. 2015-03-11]. Dostupné z: https://www.opensc-project.org/opensc/wiki/QuickStart

[31] EToken Downloads. *Isecurity.info* [online]. © 2001-2008 [cit. 2015-03-20]. Dostupné z: http://www.isecurity.info/downloads.aspx

[32] Install FreeRadius with Web Based Management Daloradius. *Linuxdrops.com* [online]. 2013 [cit. 2015-03-20]. Dostupné z: http://linuxdrops.com/install-freeradius-with-web-based-management-daloradius-on-centosrhel-debian-ubuntu/

[33] FreeRADIUS Documentation. *Freeradius.org* [online]. 2014 [cit. 2015-03-20]. Dostupné z: http://freeradius.org/doc/

[34] DaloRADIUS - download. *Sourceforge.net* [online]. 2014 [cit. 2015-03-20]. Dostupné z: http://sourceforge.net/projects/daloradius/

[35] SUNDMAN, Mathias. "ifconfig-pool" option use a /30 subnet (4 private IP addresses per client) when used in TUN mode?. *OpenVPN* [online]. © 2002-2015 [cit. 2015-03-20]. Dostupné z: http://openvpn.net/index.php/open-source/faq/77-server/273-qifconfig-poolq-option-use-a-30-subnet-4-private-ip-addresses-per-client-when-used-in-tun-mode.html

[36] Introduction to strongSwan. *Strongswan – Ipsec for Linux* [online]. © 2006-2014 [cit. 2015-04-19]. Dostupné z: https://wiki.strongswan.org/projects/strongswan/wiki/IntroductionTostrongSwan

[37] StrongSwan Installation Documentation. *Strongswan* - Ipsec for Linux [online]. © 2006-2014 [cit. 2015-04-19]. Dostupné z: https://wiki.strongswan.org/projects/strongswan/wiki/InstallationDocumentation

[38] StrongSwan on Windows. *Strongswan – Ipsec for Linux* [online]. © 2006-2014 [cit. 2015-04-19]. Dostupné z: https://wiki.strongswan.org/projects/strongswan/wiki/Windows

[39] Ipsec.conf. *Strongswan – Ipsec for Linux* [online]. © 2006-2014 [cit. 2015-04-19]. Dostupné z: https://wiki.strongswan.org/projects/strongswan/wiki/ConnSection

[40] Logger configuration. *Strongswan – Ipsec for Linux* [online]. © 2006-2014 [cit. 2015-04-19]. Dostupné z: https://wiki.strongswan.org/projects/strongswan/wiki/LoggerConfiguration

[41] IKEv2 Configuration Examples. *Strongswan – Ipsec for Linux* [online]. © 2006-2014 [cit. 2015-04-19]. Dostupné z: https://wiki.strongswan.org/projects/strongswan/wiki/IKEv2Examples

[42] StrongSwan smart card configuration HOWTO. *Strongswan – Ipsec for Linux* [online]. © 2006-2014 [cit. 2015-04-19]. Dostupné z: https://wiki.strongswan.org/projects/1/wiki/SmartCards

[43] Requirements for certificates used with Windows 7. *Strongswan – Ipsec for Linux* [online]. © 2006-2014 [cit. 2015-04-19]. Dostupné z: https://wiki.strongswan.org/projects/strongswan/wiki/Win7CertReq

[44] Windows 7 and newer. *Strongswan – Ipsec for Linux* [online]. © 2006-2014 [cit. 2015-04-19]. Dostupné z: https://wiki.strongswan.org/projects/strongswan/wiki/Windows7

[45] NetworkManager. *Strongswan – Ipsec for Linux* [online]. © 2006-2014 [cit. 2015-04-19]. Dostupné z: https://wiki.strongswan.org/projects/strongswan/wiki/NetworkManager#Smart-card-requirements

[46] IKEv1/IKEv2 Between Cisco IOS and strongSwan Configuration Example. *Strongswan – Ipsec for Linux* [online]. 2014, Jan 27 [cit. 2015-04-19]. Dostupné z: http://www.cisco.com/c/en/us/support/docs/ip/internet-key-exchange-ike/117258-config-l2l.html

Seznam příloh

Příloha A: *Generování certifikátů pomocí nástroje Easy-RSA a softwaru XCA*

Nástroj Easy RSA

V případě použití balíčku Easy RSA se jedná o nástroj určený pro vytváření a správu CA. Easy RSA je založen na OpenSSL nástrojích v prostředí příkazového řádku. Pro správnou funkci na operačním systému Ubuntu je potřeba mít nainstalovány balíky openssl a libssl-dev. Nástroj Easy RSA je u software OpenVPN 2.2.x a dřívějších verzí přibalen. Od verze OpenVPN 2.3.x je potřeba Easy RSA doinstalovat zvlášť. [26] Instalaci můžeme provést přes klasickou správu balíků (Easy RSA 2.2.2) následujícím příkazem:

```
apt-get install easy-rsa
```

Další možností je manuální instalace pomocí kompilace zdrojových kódů. Na operačním systému Windows se Easy RSA nainstaluje opět společně se softwarem OpenVPN nebo si jej lze stáhnout samostatně. Vytvořenou certifikační autoritu a vydané certifikáty můžeme použít jak pro software OpenVPN tak pro software strongSwan. Řešení strongSwan po své instalaci nabízí také například nástroj strongSwan PKI Tool (ipsec pki) více zde [27], který lze použít pro vytvoření CA a správu certifikátů taktéž.

Na operačním systému Linux si vytvoříme požadovaný adresář a do něj zkopírujeme ukázkové soubory Easy RSA. V případě že používáme Ubuntu 14.04 je zároveň nainstalován nástroj pro vytvoření požadovaných souborů, pokud máme nainstalován také software OpenVPN, může příkaz vypadat se zadanou cestou následovně:

```
make-cadir /etc/openvpn/easy-rsa
```

Nyní pomocí textového editoru (například Gedit) upravíme vytvořený soubor `vars` pomocí následujícího příkazu:

```
gedit /etc/openvpn/easy-rsa/vars
```

V souboru nastavíme základní parametry. Můžeme nastavit také například dobu platnosti certifikátů. Žádný z uvedených parametrů nesmí zůstat prázdný.

```
export KEY_COUNTRY="CZ"                 #jméno země
export KEY_PROVINCE="NA"                #jméno provincie
export KEY_CITY="Ostrava"               #jméno města
export KEY_ORG="VSB"                    #jméno organizace
export KEY_EMAIL=tab0012@vsb.cz         #email
export KEY_OU="MOB"                     #jméno organizační složky
```

Pokud používáme operační systém Windows, v prostředí příkazového řádku přejdeme do složky, kde je nástroj Easy RSA nainstalován. Nejčastěji adresář dle příkazu níže:

```
cd C:\Program Files(x86)\OpenVPN\easy-rsa
```

Následně spustíme příkaz `init-config`, kterým zahájíme konfiguraci. Dojde k nakopírování konfiguračních souborů a přepisu všech předchozích souborů. Dále upravíme dávkový soubor vars.bat pomocí poznámkového bloku stejným způsobem jako výše.

```
C:\Program Files(x86)\OpenVPN\easy-rsa>init-config
C:\Program Files(x86)\OpenVPN\easy-rsa>notepad vars.bat
```

Nyní přejdeme k počátečnímu nastavení na OS Ubuntu. Musíme být ve složce, kde se nachází Easy RSA cesta `/etc/openvpn/easy-rsa/`.

```
source vars
./clean-all
./build-dh
./pkitool --initca
```

Pomocí příkazu `build-dh` se vygenerují parametry (soubor) pro Diffie-Hellman algoritmus, který slouží k výměně certifikátů. Následujícím příkazem `pkitool --initca` vytvoříme kořenovou certifikační autoritu. Tímto se vytvoří soubory dh2048.pem a ca.crt.

Příkazy, které lze vidět níže využijeme pro vygenerování certifikátů a klíčů pro server a stranu klienta. Výstupní soubory zde budou server.crt, server.key, client.crt a client.key.

```
./pkitool --server server
./pkitool client
```

Na operačním systému Windows je postup počátečního nastavení a generování potřebných souborů stejný jako na operačním systému Ubuntu. Je zde pouze trochu rozdílná syntaxe a příkaz `source vars` se zde zadává pouze jako `vars`. Níže lze vidět rozdíl v syntaxi, kdy se vynechají v příkazech znaky „./”. [24]

```
./clean-all     #Ubuntu          clean-all       #Windows
./build-dh      #Ubuntu          build-ca        #Windows
```

Nástroj Easy RSA obsahuje také další sadu příkazů, příklady použití a kompletní seznam můžeme nalézt v přiloženém textovém souboru „readme”. Certifikáty a klíče je možné také generovat například do formátu .pem nebo .p12 nebo rušit platnost certifikátů. Veškeré uvedené příkazy výše se mohou lišit dle použité verze softwaru. Easy RSA je jednoduchý nástroj pro vytváření CA a následné generování certifikátů, v případě většího počtu klientů je ale vhodné pro správu využít více efektivní a uživatelsky přívětivější software, například XCA.

Software XCA

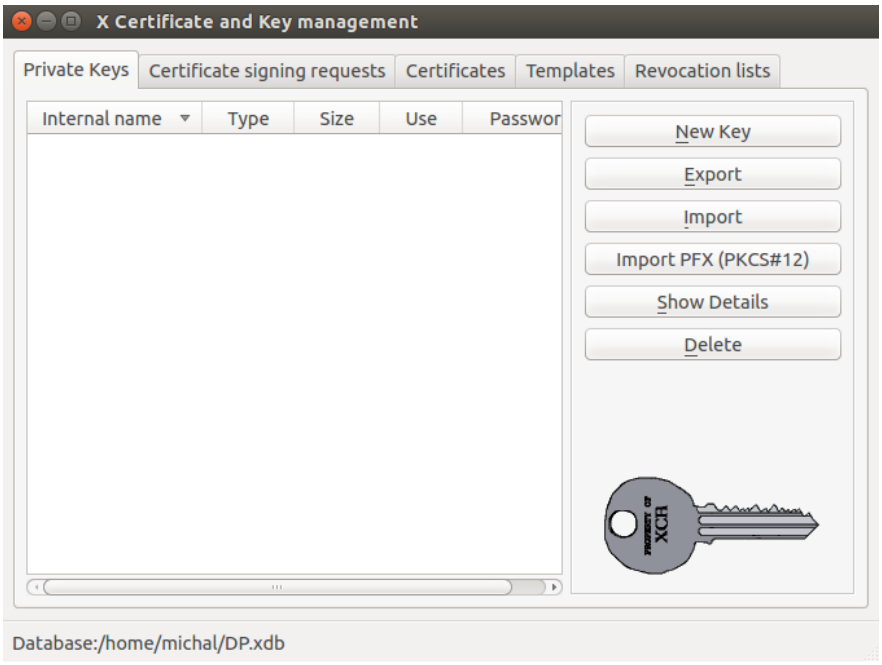
Software XCA (X Certificate and key management) využívá pro svou práci kryptografické knihovny OpenSSL. Oproti samotnému OpenSSL ale nabízí grafické uživatelské rozhraní. XCA je volně dostupný software, který podporuje operační systémy Linux, Windows a Mac OS. Z domovské stránky [28] lze stáhnout všechny potřebné soubory pro instalaci. Na

operačním systému Ubuntu 14.04 LTS doporučuji instalaci pomocí klasické správy balíků následujícím příkazem:

```
apt-get install xca
```

Pro vytvoření certifikační autority a certifikátů pro server a klienty můžeme využít právě tento software. Pomoci XCA lze vytvořit CA, generovat a spravovat soukromé klíče algoritmů RSA a DSA o velikosti až 4096 bitů. Dále lze generovat certifikáty, vytvářet žádosti o certifikát, zneplatnit existující certifikát a vygenerovat seznam zneplatněných certifikátů (CRL). Veškerá generovaná data se ukládají do databázového souboru, který může být chráněn heslem. Do programu můžeme také importovat již vydané certifikáty, nechybí ani export vytvořených certifikátu v různých formátech. Výhodnou vlastností je podpora pro hardwarová uložiště prostřednictvím rozhraní PKCS#11. Pomocí XCA lze tedy inicializovat USB token, nastavit hesla, ukládat na něj klíče a certifikáty. Právě tyto vlastnosti využijeme v další kapitole pro práci s USB tokenem. Dále se krátce věnuji vytvoření certifikační autority a vygenerování soukromých klíčů a certifikátů (XCA verze 0.9.3).

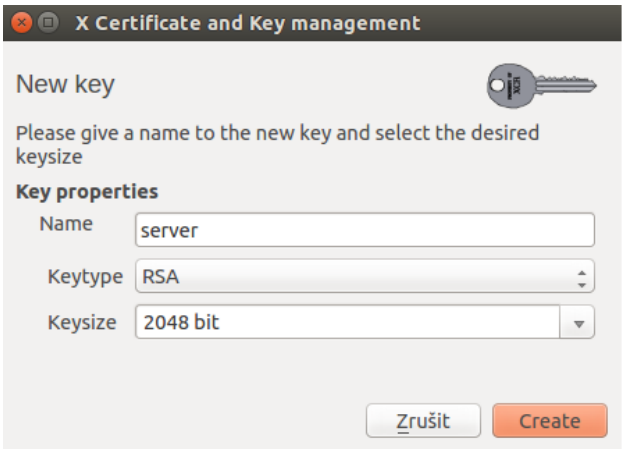
Po spuštění programu je potřeba vytvořit nový databázový soubor (nebo otevřít existující), který slouží k uložení klíčů a certifikátů. K tomuto slouží v menu položka „File" a následně „New DataBase". Vybereme umístění databáze, můžeme také zvolit heslo pro ochranu dat. Na obrázku níže lze vidět hlavní okno XCA. Vše je přehledně členěno v samostatných záložkách (klíče, žádosti o certifikát, certifikáty, šablony, revokační listy).



XCA - Úvodní okno programu

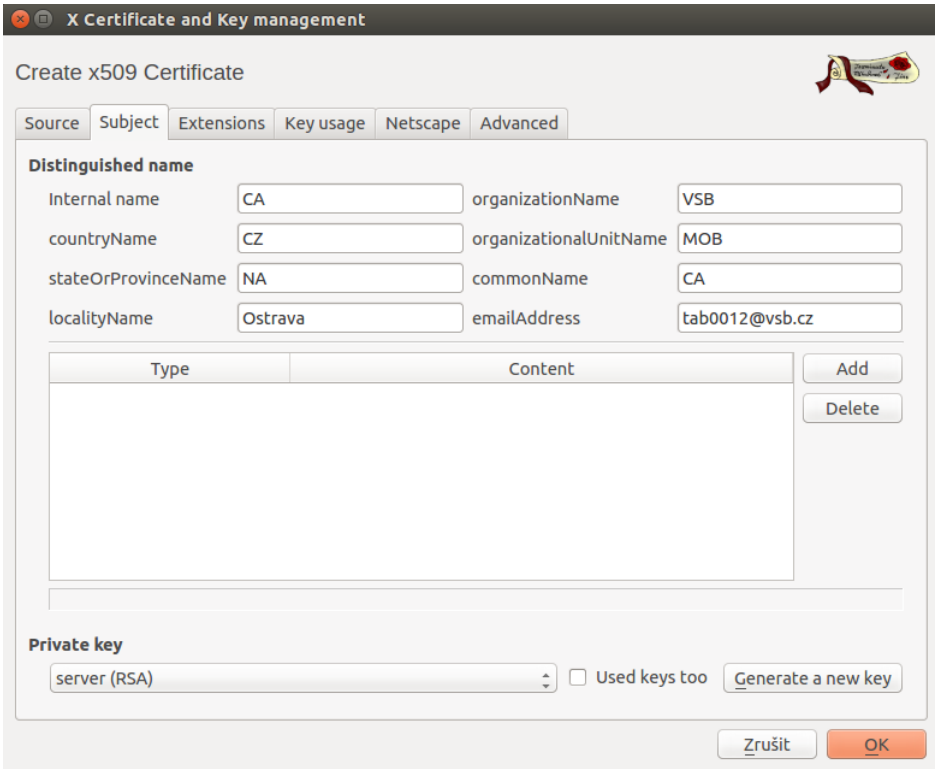
Při vygenerování certifikátu CA nebo následné vydávání certifikátů si můžeme nejprve potřebné soukromé klíče připravit a vygenerovat na záložce „Private Keys" nebo potřebné klíče generovat při vytváření certifikátu ve formuláři během tvorby. Při tvorbě soukromého klíče

zadáme jeho název, vybereme algoritmus klíče (RSA) a velikost klíče (obvykle 2048 bitů). Vytvoření klíče lze vidět na následujícím obrázku.



XCA - Vytváření soukromého klíče

Pro vytvoření certifikační autority klikneme na záložku „Certificates" a zde stiskneme tlačítko „New Certificate", následně se zobrazí okno, pomocí kterého specifikujeme strukturu certifikátu CA. Nyní je popsáno základní nastavení pro vytvoření CA. Na první záložce „Source" vybereme parametr „Template for the new certificate" jako „default CA" (vytvoření vlastní certifikační autority) stiskneme tlačítko „Apply all", které nám ze šablony vyplní některé parametry na dalších záložkách.



XCA - Vytváření certifikátu CA (záložka Subject)

Na další záložce „Subject“ zkontrolujeme a vyplníme požadované informace (podobně jako u Easy RSA v souboru vars), dále zvolíme potřebný soukromý klíč nebo vygenerujeme nový. Na záložce „Extensions“ nastavíme rozšíření certifikátu. Parametr „Type“ nastavíme na „Certification Authority“, pomocí parametru „Path Length“ určíme, kolik lze vytvořit pod touto CA podřízených certifikačních autorit. Zatrhneme políčka „Subject Key Identifier“ a „Authority Key Identifier“. Také můžeme nastavit dobu platnosti CA. Další záložky nastavíme podle svých potřeb nebo necháme v defaultním nastavení, vše potvrdíme tlačítkem OK a vytvoříme CA.

Generování uživatelského certifikátu je velice podobné předchozímu postupu. Opět na záložce „Certificates“ klikneme na „New Certificate“. Na záložce „Source“ u parametru „Use this Certificate for signing“ vybereme certifikát CA, kterým podepíšeme vygenerovaný certifikát. U položky „Template for the new certificate“ můžeme vybrat šablonu pro server nebo klienta. Na druhé záložce „Subject“ vyplníme požadované informace a vybereme soukromý klíč. Na záložce „Extensions“ u parametru „Type“ vybereme typ „End Entity“. Zatrhneme obě políčka „Key identifier“. Dále lze nastavit délku platnosti vydávaného certifikátu. Další záložky necháme v defaultním nastavení nebo nakonfigurujme podle svých potřeb. Na záložce „Key usage“ ovšem doporučuji u parametru „Extended key usage“ zatrhnout volbu „TLS Web Server Authentication“ nebo „TLS Web Client Authentication“ podle toho, zda generujeme certifikát pro server nebo pro klienta. Tato volba rozšířeného využití klíče je u některých aplikací pro správné navázání VPN spojení požadována. Vše potvrdíme tlačítkem OK a vytvoříme certifikát pro server nebo požadované klienty. Pomocí XCA lze také vygenerovat soubor s parametry pro Diffie Hellman algoritmus, k tomuto slouží v menu položka „File“ a následně „Generate DH parametr“. [28]

Příloha B: *Technické specifikace - Aladdin eToken PRO 32k*

Technické specifikace	
Produktové jméno	eToken PRO 32K 4.2B
Model	Token 4.25 1.2 2.6.189
Podporované operační systémy	**Windows:** Windows 8.1 (32-bit a 64-bit), Windows Server 2012 R2 a všechny předešlé verze. **Mac:** Mac OS X 10.9 (Mavericks) a všechny předešlé verze. **Linux:** Red Hat, Ubuntu, Debian, SUSE, CentOS a Fedora.
API a standartní podpora	PKCS#11 V2.20, MS CryptoAPI a CNG (CSP, KSP), Mac Keychain (TokenD), Java Card 2.2.2, GlobalPlatform Verze 2.1.1, PC/SC, X.509 v3, SSL v3, IPSec/IKE
Bezpečnostní algoritmy	RSA 1024-bit / 2048-bit, DES, 3DES, SHA1
Rozměry	52 x 16 x 8 mm
USB konektor	USB typ A, podpora USB 1.1 a 2.0
Voděodolnost	IP X7 – IEC 529
Délka uchování dat	Nejméně 10 let
Počet přepisů paměti	Nejméně 500,000x
Voděodolnost	IP X7 – IRC 529
Pracovní teplota	0°C až 70°C
Skladovací teplota	-40°C až 85°C
Velikost paměti	32 768 kB
Volná paměť	25 077 kB
Jméno ovladače	AKS ifdh 0
Verze firmware	2.6
Verze hardware	4.25
Typ karty	CardOS
Verze OS	CardOS V4.2B © Siemens AG 1994-2005

Příloha C: *Výpis nástroje pcsc_scan*

```
PC/SC device scanner
V 1.4.22 (c) 2001-2011, Ludovic Rousseau
<ludovic.rousseau@free.fr>
Compiled with PC/SC lite version: 1.8.10
Using reader plug'n play mechanism
Scanning present readers...
0: Aladdin eToken PRO 64 00 00

Thu Mar  5 18:54:16 2015
Reader 0: Aladdin eToken PRO 64 00 00
  Card state: Card inserted,
  ATR: 3B F2 18 00 02 C1 0A 31 FE 58 C8 09 75

ATR: 3B F2 18 00 02 C1 0A 31 FE 58 C8 09 75
+ TS = 3B --> Direct Convention
+ T0 = F2, Y(1): 1111, K: 2 (historical bytes)
  TA(1) = 18 --> Fi=372, Di=12, 31 cycles/ETU
    129032 bits/s at 4 MHz, fMax for Fi = 5 MHz => 161290 bits/s
  TB(1) = 00 --> VPP is not electrically connected
  TC(1) = 02 --> Extra guard time: 2
  TD(1) = C1 --> Y(i+1) = 1100, Protocol T = 1
-----
  TC(2) = 0A --> Work waiting time: 960 x 10 x (Fi/F)
  TD(2) = 31 --> Y(i+1) = 0011, Protocol T = 1
-----
  TA(3) = FE --> IFSC: 254
  TB(3) = 58 --> Block Waiting Integer: 5 - Character Waiting
Integer: 8
+ Historical bytes: C8 09
  Category indicator byte: C8 (proprietary format)
+ TCK = 75 (correct checksum)

Possibly identified card (using
/usr/share/pcsc/smartcard_list.txt):
3B F2 18 00 02 C1 0A 31 FE 58 C8 09 75
           Siemens CardOS V4.2B
```

Příloha D: *Výpis objektů na tokenu – příkaz pkcs15-tool –dump*

```
Using reader with a card: Aladdin eToken PRO 64 00 00
PKCS#15 Card [OpenSC Card]:
      Version        : 0
      Serial number  : 26564319142F
      Manufacturer ID: OpenSC Project
      Last update    : 20150306103433Z
      Flags          : EID compliant

PIN [Security Officer PIN]
      Object Flags   : [0x3], private, modifiable
      ID             : ff
      Flags          : [0xB2], local, initialized, soPin
      Length         : min_len:6, max_len:8, stored_len:8
      Pad char       : 0x00
      Reference      : 1 (0x01)
      Type           : ascii-numeric
      Path           : 3f005015

PIN [Michal]
      Object Flags   : [0x3], private, modifiable
      ID             : 01
      Flags          : [0x32], local, initialized, needs-padding
      Length         : min_len:4, max_len:8, stored_len:8
      Pad char       : 0x00
      Reference      : 3 (0x03)
      Type           : ascii-numeric
      Path           : 3f005015

Private RSA Key [Certificate]
      Object Flags   : [0x3], private, modifiable
      Usage          : [0x4], sign
      Access Flags   : [0xD], sensitive, alwaysSensitive,
neverExtract
      ModLength      : 2048
      Key ref        : 16 (0x10)
      Native         : yes
      Path           : 3f005015
      Auth ID        : 01
      ID             : 45
      GUID           : {7f402dda-a40b-4bbd-88b7-d33dd9cbd969}
```

```
Public RSA Key [Public Key]
      Object Flags   : [0x2], modifiable
      Usage          : [0x40], verify
      Access Flags   : [0x0]
      ModLength      : 2048
      Key ref        : 0 (0x0)
      Native         : no
      Path           : 3f0050153003
      ID             : 45
      DirectValue    : <absent>

X.509 Certificate [Certificate]
      Object Flags   : [0x2], modifiable
      Authority      : no
      Path           : 3f0050153104
      ID             : 45
      GUID           : {7f402dda-a40b-4bbd-88b7-d33dd9cbd969}
      Encoded serial : 02 01 02
```

Příloha E: *Tabulka nainstalovaných balíků a softwaru pro práci s tokenem*

Operační systém Ubuntu	
Název	**Verze**
OS Ubuntu	14.04 LTS (32 bit a 64 bit)
XCA	0.9.3
Easy-rsa	2.2.2
eToken PKI Client	5.00.28-0 a 5.00.59-0
OpenSC	0.13.0
OpenCT	0.6.20
Pcscd	1.8.10
Pcsc-tools	1.4.22
Libhal1	0.5.14
Libpcsclite1	1.8.10
Libccid	1.4.15
Libqt4-core	4:4.8.5
Libqt4-gui	4:4.8.5
Alien	8.90
OpenSSL	1.0.1f
Operační systém Windows	
OS Windows	Windows 7 (64 bit)
eToken PKI Client - Windows	5.1 SP1 (5.1.66.0)
XCA	1.1.0

Příloha F: *OpenVPN ukázka – základní konfigurační soubory*

Soubor serveru /etc/openvpn/server.conf.

```
dev tun
ifconfig 10.0.0.1 10.0.0.2
proto udp
port 1194
auth SHA256
cipher AES-256-CBC
secret /etc/openvpn/klic.key
copm-lzo
user nobody
group nogroup
verb 3
```

Soubor klienta /etc/openvpn/klient.conf**.**

```
dev tun
remote 192.168.1.108
ifconfig 10.0.0.2 10.0.0.1
proto udp
port 1194
auth SHA256
cipher AES-256-CBC
secret /etc/openvpn/klic.key
copm-lzo
user nobody
group nogroup
verb 3
```

Příloha G: *OpenVPN – konfigurace serveru a klienta*

Konfigurace serveru

Soubor /etc/openvpn/server.conf.

```
tls-server
mode server
server 10.0.0.0 255.255.255.0
push "route 192.168.1.0 255.255.255.0"
dev tun
proto udp
port 1194
keepalive 10 120
cipher AES-256-CBC
auth SHA1
ca ca.crt
cert server.crt
key server.key
dh dh2048.pem
tls-auth ta.key 0
comp-lzo
verb 3
max-clients 10
user nobody
group nogroup
persist-tun
persist-key
#plugin /usr/lib/openvpn/openvpn-plugin-auth-pam.so login
plugin /etc/openvpn/radiusplugin.cnf /etc/openvpn/radiusplugin.so
```

Potřebné soubory v konfiguraci:

```
/etc/openvpn/radiusplugin.cnf
/etc/openvpn/radiusplugin.so
```

Konfigurace serveru

Soubor /etc/openvn/klient.conf

```
tls-client
pull
remote 10.10.10.30
nobind
cipher AES-256-CBC
auth SHA1
dev tun
proto udp
port 1194
ca ca.crt
#cert klient.crt
#key klient.key
tls-auth ta.key 1
comp-lzo
user nobody
group nogroup
presist-tun
presist-key
verb 3
auth-nocache
auth-user-pass
explicit-exit-notify 3
resolv-retry infinite
remote-cert-tls server
verify-x509-name server name
pkcs11-providers /usr/lib64/libeTPkcs11.so
pkcs11-id 'Aladdin\x20Ltd\x2E/eToken/26565946222f/Aladdin/7F558'
```

```
root@ubuntu:/home/michal# openvpn --show-pkcs11-ids
/usr/lib64/libeTPkcs11.so


The following objects are available for use.

Each object shown below may be used as parameter to

--pkcs11-id option please remember to use single quote mark.


Certificate

       DN:             C=CZ, ST=NA, L=Ostrava, O=VSB, OU=MOB,
CN=klient, name=DP, emailAddress=tab0012@vsb.cz

       Serial:         02

       Serialized id:
Aladdin\x20Ltd\x2E/eToken/26564319142f/Aladdin/7FF286DDB11116
```

Příloha H: *Konfigurace MySQL databáze*

Nainstalované balíky – freerafius, freeradius-mysql, freeradius-utils, mysql-server. Během instalace MySQL, zadáme heslo pro databázi, níže spuštění služby MySQL.

```
/etc/init.d/mysql start
```

Nyní musíme vytvořit databázi radius, přihlásíme se k MySQL serveru.

```
mysql –uroot -p
```

Vytvoření databáze s názvem radius a udělení oprávnění.

```
mysql> CREATE DATABASE radius;

mysql> GRANT ALL PRIVILEGES ON radius.* TO radius@localhost
IDENTIFIED BY "password";

mysql> flush privileges;

mysql> exit
```

V posledním kroku naplníme databázi, kterou jsme vytvořili s připraveným schématem. Schéma se nainstalovalo během instalace serveru FreeRADIUS.

```
mysql -uradius -p radius < /etc/freeradius/sql/mysql/schema.sql
```

Nyní upravíme soubor sql.conf, níže nastavení požadovaných parametrů, zbytek konfigurace necháme v defaultním nastavení.

```
gedit /etc/freeradius/sql.conf


#Connection info:

server = "localhost"

port = 3306

login = "radius"

password = "password"

radius_db = "radius"

acct_table1 = "radacct"

acct_table2 = "radacct"

authcheck_table = "radcheck"

authreply_table = "radreply"

groupcheck_table = "radgroupcheck"

groupreply_table = "radgroupreply"

usergroup_table = "radusergroup"
```

Příloha I: *Konfigurace aplikace daloRADIUS*

Postup konfigurace

Nainstalované balíky – php5, php5-mysql, php5-gd, php-pear, php-db. Následné rozbalení staženého souboru daloRADIUS pomocí příkazu `tar xzf`.

```
tar xzf daloradius-0.9-9.tar.gz
```

Importování tabulek do databáze.

```
mysql -uradius -p radius < daloradius-0.9-9/contrib/db/fr2-
mysql-daloradius-and-freeradius.sql
```

Nyní provedeme úpravy v konfiguračním souboru daloRADIUS.

```
gedit daloradius-0.9-9/library/daloradius.conf.php
```

Nastavené parametry:

```
$configValues['DALORADIUS_VERSION'] = '0.9-9';
$configValues['FREERADIUS_VERSION'] = '2';
$configValues['CONFIG_DB_ENGINE'] = 'mysql';
$configValues['CONFIG_DB_HOST'] = 'localhost';
$configValues['CONFIG_DB_USER'] = 'radius';
$configValues['CONFIG_DB_PASS'] = 'password';
$configValues['CONFIG_DB_NAME'] = 'radius';
```

Přesunutí daloradius do kořenového adresáře webu.

```
mv daloradius-0.9-9 /var/www/html/daloradius
```

Přiřazení práv chmod 755 pro následující soubory a adresáře.

```
/var/log/syslog
/var/log/dmesg
/var/log/freeradius –R
/var/www/html/daloradius -R
```

Webové rozhraní daloRADIUS je v tomto případě dostupné na:

```
http://10.10.10.20/daloradius
```

Defaultní přihlašovací údaje:

```
Username: administrator
Password: radius
```

Příloha J: *Virtuální rozhraní serveru a klienta - výpis*

Virtuální rozhraní serveru.

```
tun0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          inet addr:10.0.0.1  P-t-P:10.0.0.2  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:61 errors:0 dropped:0 overruns:0 frame:0
          TX packets:61 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:5124 (5.1 KB)  TX bytes:5124 (5.1 KB)
```

Virtuální rozhraní klienta.

```
tun0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          inet addr:10.0.0.6  P-t-P:10.0.0.5  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:4 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:336 (336.0 B)  TX bytes:336 (336.0 B)
```

Pro více informací o virtuálním rozhraní a způsobu přidělování virtuálních adres, doporučuju stránku [35] na oficiálním webu OpenVPN.

Příloha K: *OpenVPN - výpisy vytváření spojení a výpis z Wiresharku*

Výpisy ze strany serveru

Výpis při spuštění serveru.

```
Tue Mar 10 12:25:05 2015 OpenVPN 2.3.2 x86_64-pc-linux-gnu [SSL
(OpenSSL)] [LZO] [EPOLL] [PKCS11] [eurephia] [MH] [IPv6] built
on Dec 1 2014

Tue Mar 10 12:25:05 2015 RADIUS-PLUGIN: Configfile name:
/etc/openvpn/radiusplugin.cnf.

Tue Mar 10 12:25:05 2015 PLUGIN_INIT: POST
/etc/openvpn/radiusplugin.so '[/etc/openvpn/radiusplugin.so]
[/etc/openvpn/radiusplugin.cnf]'intercepted=PLUGIN_AUTH_USER_PAS
S_VERIFY|PLUGIN_CLIENT_CONNECT|PLUGIN_CLIENT_DISCONNECT

Tue Mar 10 12:25:05 2015 Diffie-Hellman initialized with 2048
bit key

Tue Mar 10 12:25:05 2015 Control Channel Authentication: using
'ta.key' as a OpenVPN static key file

Tue Mar 10 12:25:05 2015 Outgoing Control Channel
Authentication: Using 160 bit message hash 'SHA1' for HMAC
authentication

Tue Mar 10 12:25:05 2015 Incoming Control Channel
Authentication: Using 160 bit message hash 'SHA1' for HMAC
authentication

Tue Mar 10 12:25:05 2015 Socket Buffers: R=[87380->131072]
S=[16384->131072]

Tue Mar 10 12:25:05 2015 ROUTE_GATEWAY 10.10.10.1/255.255.255.0
IFACE=eth0 HWADDR=74:d4:35:7c:71:8c

Tue Mar 10 12:25:05 2015 TUN/TAP device tun0 opened

Tue Mar 10 12:25:05 2015 TUN/TAP TX queue length set to 100

Tue Mar 10 12:25:05 2015 do_ifconfig, tt->ipv6=0,

tt->did_ifconfig_ipv6_setup=0

Tue Mar 10 12:25:05 2015 /sbin/ip link set dev tun0 up mtu 1500

Tue Mar 10 12:25:05 2015 /sbin/ip addr add dev tun0 local
10.0.0.1 peer 10.0.0.2

Tue Mar 10 12:25:05 2015 /sbin/ip route add 10.0.0.0/24 via
10.0.0.2

Tue Mar 10 12:25:05 2015 GID set to nogroup
```

```
Tue Mar 10 12:25:05 2015 UID set to nobody

Tue Mar 10 12:25:05 2015 UDPv4 link local (bound):[undef]

Tue Mar 10 12:25:05 2015 UDPv4 link remote: [undef]

Tue Mar 10 12:25:05 2015 MULTI: multi_init called, r=256 v=256

Tue Mar 10 12:25:05 2015 IFCONFIG POOL: base=10.0.0.4 size=62,
ipv6=0

Tue Mar 10 12:25:05 2015 Initialization Sequence Completed
```

Výpis při připojení klienta:

```
Tue Mar 10 12:25:24 2015 20.20.20.10:38224 TLS: Initial packet
from [AF_INET]20.20.20.10:38224, sid=cc25f24b 86f111f2

Tue Mar 10 12:25:30 2015 20.20.20.10:38224 VERIFY OK: depth=1,
C=CZ, ST=NA, L=Ostrava, O=VSB, OU=MOB, CN=VSB CA, name=DP,
emailAddress=tab0012@vsb.cz

Tue Mar 10 12:25:30 2015 20.20.20.10:38224 VERIFY OK: depth=0,
C=CZ, ST=NA, L=Ostrava, O=VSB, OU=MOB, CN=klient, name=DP,
emailAddress=tab0012@vsb.cz

Tue Mar 10 12:25:30 2015 RADIUS-PLUGIN: FOREGROUND THREAD:
Auth_user_pass_verify thread started.

Tue Mar 10 12:25:30 2015 RADIUS-PLUGIN: FOREGROUND THREAD: New
user.

Tue Mar 10 12:25:30 2015 RADIUS-PLUGIN: FOREGROUND THREAD: Add
user to map.

Tue Mar 10 12:25:30 2015 20.20.20.10:38224 PLUGIN_CALL: POST
/etc/openvpn/radiusplugin.so/PLUGIN_AUTH_USER_PASS_VERIFY
status=0

Tue Mar 10 12:25:30 2015 20.20.20.10:38224 TLS:
Username/Password authentication succeeded for username 'klient'

Tue Mar 10 12:25:30 2015 20.20.20.10:38224 Data Channel Encrypt:
Cipher 'AES-256-CBC' initialized with 256 bit key

Tue Mar 10 12:25:30 2015 20.20.20.10:38224 Data Channel Encrypt:
Using 160 bit message hash 'SHA1' for HMAC authentication

Tue Mar 10 12:25:30 2015 20.20.20.10:38224 Data Channel Decrypt:
Cipher 'AES-256-CBC' initialized with 256 bit key

Tue Mar 10 12:25:30 2015 20.20.20.10:38224 Data Channel Decrypt:
Using 160 bit message hash 'SHA1' for HMAC authentication
```

```
Tue Mar 10 12:25:30 2015 20.20.20.10:38224 Control Channel:
TLSv1, cipher TLSv1/SSLv3 DHE-RSA-AES256-SHA, 2048 bit RSA

Tue Mar 10 12:25:30 2015 20.20.20.10:38224 [klient] Peer
Connection Initiated with [AF_INET]20.20.20.10:38224

Tue Mar 10 12:25:30 2015 klient/20.20.20.10:38224 MULTI_sva:
pool returned IPv4=10.0.0.6, IPv6=(Not enabled)

Tue Mar 10 12:25:30 2015 klient/20.20.20.10:38224 PLUGIN_CALL:
POST /etc/openvpn/radiusplugin.so/PLUGIN_CLIENT_CONNECT status=0

Tue Mar 10 12:25:30 2015 klient/20.20.20.10:38224 OPTIONS
IMPORT: reading client specific options from:
/tmp/openvpn_cc_5cb44708719ba23928abee76dc28a263.tmp

Tue Mar 10 12:25:30 2015 klient/20.20.20.10:38224 MULTI: Learn:
10.0.0.6 -> klient/20.20.20.10:38224

Tue Mar 10 12:25:30 2015 klient/20.20.20.10:38224 MULTI: primary
virtual IP for klient/20.20.20.10:38224: 10.0.0.6

Tue Mar 10 12:25:32 2015 klient/20.20.20.10:38224 PUSH: Received
control message: 'PUSH_REQUEST'

Tue Mar 10 12:25:32 2015 klient/20.20.20.10:38224
send_push_reply(): safe_cap=940

Tue Mar 10 12:25:32 2015 klient/20.20.20.10:38224 SENT CONTROL
[klient]: 'PUSH_REPLY,route 192.168.1.0 255.255.255.0,route
10.0.0.1,topology net30,ping 10,ping-restart 120,ifconfig
10.0.0.6 10.0.0.5' (status=1)
```

Výpisy ze strany klienta

Výpis spuštění a připojení k serveru.

```
Tue Mar 10 12:29:41 2015 OpenVPN 2.3.2 x86_64-pc-linux-gnu [SSL
(OpenSSL)] [LZO] [EPOLL] [PKCS11] [eurephia] [MH] [IPv6] built
on Dec  1 2014

Enter Auth Username:klient

Enter Auth Password:

Tue Mar 10 12:29:49 2015 PKCS#11: Adding PKCS#11 provider
'/usr/lib64/libeTPkcs11.so'

Tue Mar 10 12:29:50 2015 Control Channel Authentication: using
'ta.key' as a OpenVPN static key file
```

```
Tue Mar 10 12:29:50 2015 Outgoing Control Channel
Authentication: Using 160 bit message hash 'SHA1' for HMAC
authentication

Tue Mar 10 12:29:50 2015 Incoming Control Channel
Authentication: Using 160 bit message hash 'SHA1' for HMAC
authentication

Tue Mar 10 12:29:50 2015 Socket Buffers: R=[87380->131072]
S=[16384->131072]

Tue Mar 10 12:29:50 2015 NOTE: UID/GID downgrade will be delayed
because of --client, --pull, or --up-delay

Tue Mar 10 12:29:51 2015 UDPv4 link local: [undef]

Tue Mar 10 12:29:51 2015 UDPv4 link remote:
[AF_INET]10.10.10.30:1194

Tue Mar 10 12:29:51 2015 TLS: Initial packet from
[AF_INET]10.10.10.30:1194, sid=d3999531 9d5cada2

Tue Mar 10 12:29:51 2015 VERIFY OK: depth=1, C=CZ, ST=NA,
L=Ostrava, O=VSB, OU=MOB, CN=VSB CA, name=DP,
emailAddress=tab0012@vsb.cz

Tue Mar 10 12:29:51 2015 Validating certificate key usage

Tue Mar 10 12:29:51 2015 ++ Certificate has key usage 00a0,
expects 00a0

Tue Mar 10 12:29:51 2015 VERIFY KU OK

Tue Mar 10 12:29:51 2015 Validating certificate extended key
usage

Tue Mar 10 12:29:51 2015 ++ Certificate has EKU (str) TLS Web
Server Authentication, expects TLS Web Server Authentication

Tue Mar 10 12:29:51 2015 VERIFY EKU OK

Tue Mar 10 12:29:51 2015 VERIFY X509NAME OK: C=CZ, ST=NA,
L=Ostrava, O=VSB, OU=MOB, CN=server, name=DP,
emailAddress=tab0012@vsb.cz

Tue Mar 10 12:29:51 2015 VERIFY OK: depth=0, C=CZ, ST=NA,
L=Ostrava, O=VSB, OU=MOB, CN=server, name=DP,
emailAddress=tab0012@vsb.cz

Enter Aladdin token Password:

Tue Mar 10 12:29:57 2015 Data Channel Encrypt: Cipher 'AES-256-
CBC' initialized with 256 bit key

Tue Mar 10 12:29:57 2015 Data Channel Encrypt: Using 160 bit
message hash 'SHA1' for HMAC authentication
```

```
Tue Mar 10 12:29:57 2015 Data Channel Decrypt: Cipher 'AES-256-
CBC' initialized with 256 bit key

Tue Mar 10 12:29:57 2015 Data Channel Decrypt: Using 160 bit
message hash 'SHA1' for HMAC authentication

Tue Mar 10 12:29:57 2015 Control Channel: TLSv1, cipher
TLSv1/SSLv3 DHE-RSA-AES256-SHA, 2048 bit RSA

Tue Mar 10 12:29:57 2015 [server] Peer Connection Initiated with
[AF_INET]10.10.10.30:1194

Tue Mar 10 12:29:59 2015 SENT CONTROL [server]: 'PUSH_REQUEST'
(status=1)

Tue Mar 10 12:30:00 2015 PUSH: Received control message:
'PUSH_REPLY,route 192.168.1.0 255.255.255.0,route
10.0.0.1,topology net30,ping 10,ping-restart 120,ifconfig
10.0.0.6 10.0.0.5'

Tue Mar 10 12:30:00 2015 OPTIONS IMPORT: timers and/or timeouts
modified

Tue Mar 10 12:30:00 2015 OPTIONS IMPORT: --ifconfig/up options
modified

Tue Mar 10 12:30:00 2015 OPTIONS IMPORT: route options modified

Tue Mar 10 12:30:00 2015 ROUTE_GATEWAY 20.20.20.1/255.255.255.0
IFACE=eth0 HWADDR=74:d4:35:7c:29:ed

Tue Mar 10 12:30:00 2015 TUN/TAP device tun0 opened

Tue Mar 10 12:30:00 2015 TUN/TAP TX queue length set to 100

Tue Mar 10 12:30:00 2015 do_ifconfig, tt->ipv6=0,

tt->did_ifconfig_ipv6_setup=0

Tue Mar 10 12:30:00 2015 /sbin/ip link set dev tun0 up mtu 1500

Tue Mar 10 12:30:00 2015 /sbin/ip addr add dev tun0 local
10.0.0.6 peer 10.0.0.5

Tue Mar 10 12:30:00 2015 /sbin/ip route add 192.168.1.0/24 via
10.0.0.5

Tue Mar 10 12:30:00 2015 /sbin/ip route add 10.0.0.1/32 via
10.0.0.5

Tue Mar 10 12:30:00 2015 Initialization Sequence Complete
```

Výpis z Wiresharku

Ukázka obsahu zprávy Client Hello. Jedná se pouze o ukázku, zprávy obsahují jinak velké množství informací.

```
⊞ OpenVPN Protocol
⊟ Secure Sockets Layer
  ⊟ TLSv1 Record Layer: Handshake Protocol: Client Hello
      Content Type: Handshake (22)
      Version: TLS 1.0 (0x0301)
      Length: 208
    ⊟ Handshake Protocol: Client Hello
        Handshake Type: Client Hello (1)
        Length: 204
        Version: TLS 1.0 (0x0301)
      ⊞ Random
        Session ID Length: 0
        Cipher Suites Length: 90
      ⊞ Cipher Suites (45 suites)
        Compression Methods Length: 1
      ⊞ Compression Methods (1 method)
        Extensions Length: 73
      ⊞ Extension: ec_point_formats
      ⊞ Extension: elliptic_curves
      ⊞ Extension: SessionTicket TLS
      ⊞ Extension: Heartbeat
```

Ukázka obsahu zprávy Server Hello.

```
⊞ OpenVPN Protocol
⊟ Secure Sockets Layer
  ⊟ TLSv1 Record Layer: Handshake Protocol: Server Hello
      Content Type: Handshake (22)
      Version: TLS 1.0 (0x0301)
      Length: 58
    ⊟ Handshake Protocol: Server Hello
        Handshake Type: Server Hello (2)
        Length: 54
        Version: TLS 1.0 (0x0301)
      ⊞ Random
        Session ID Length: 0
        Cipher Suite: TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x0039)
        Compression Method: null (0)
        Extensions Length: 14
      ⊞ Extension: renegotiation_info
      ⊞ Extension: SessionTicket TLS
      ⊞ Extension: Heartbeat
  ⊞ TLSv1 Record Layer: Handshake Protocol: Certificate
  ⊞ TLSv1 Record Layer: Handshake Protocol: Server Key Exchange
  ⊞ TLSv1 Record Layer: Handshake Protocol: Multiple Handshake M
```

Příloha L: *Konfigurace logování v strongswan.conf*

```
charon {
    #definování logu do souboru
    filelog {
        /var/log/charon.log {
            # přepsání existujícího souboru
            append = no
            #zvýšení defaultní hodnoty pro všechny subsystémy
            default = 2
            ike = 2
            knl = 3
        }
    }
    # log využívající syslog
    syslog {
        # prefix pro každou zapsanou zprávu
        identifier = charon-custom
        #minimalistický IKE audit log
        auth {
            default = -1
            ike = 0
        }
    }
}
```

Příloha M: *StrongSwan – konfigurace serveru a klienta*

Konfigurace serveru

/etc/ipsec.conf

```
conn %default
      ikelifetime=60m
      keylife=20m
      keyingtries=1
      keyexchange=ikev2
      ike=aes256-sha1-modp1024!
      esp=aes256-sha1!
conn VPN-tunel
      left=10.10.10.30
      leftcert=server.crt
      leftsubnet=192.168.1.0/24
      leftid="C=CZ, ST=NA, L=Ostrava, O=VSB, OU=MOB, CN=server,
      N=DP, E=tab0012@vsb.cz"
      leftfirewall=yes
      leftauth=pubkey
      rightauth=eap-radius
      rightsourceip=10.0.0.0/24
      right=%any
      auto=add
```

/etc/ipsec.secret

```
: RSA server.key
```

/etc/strongswan.d/charon/eap-radius.conf

```
eap-radius {
      server = 10.10.10.20
      secret = 123456
    }
```

/etc/strongswan.conf

```
charon {
        load_modular = yes
        pugins {
                include strongswan.d/charon/*.conf
        }
}
include strongswan.d/*.conf
```

Nastavení iptables pravidel.

```
iptables -A FORWARD -i eth0 -j ACCEPT
iptables -A FORWARD -i eth1 -j ACCEPT
```

```
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  192.168.0.100        10.1.0.0/16
policy match dir in pol ipsec reqid 1 proto 50
ACCEPT     all  --  10.1.0.0/16          192.168.0.100
policy match dir out pol ipsec reqid 1 proto 50

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Konfigurace klienta

/etc/ipsec.conf

```
conn %default
      ikelifetime=60m
      keylife=20m
      keyingtries=1
```

```
      keyexchange=ikev2
      ike=aes256-sha1-modp1024!
      esp=aes256-sha1!
conn VPN-tunel
      left=20.20.20.10
      leftcert=%smartcard:6FF32AB18778BCABBB509E597CCAF3E7817E21B
      #leftid="C=CZ, ST=NA, L=Ostrava, O=VSB, OU=MOB, CN=klient,
      N=DP, E=tab0012@vsb.cz"
      leftid=klient
      leftfirewall=yes
      leftsourceip=%config
      leftauth=eap
      rightauth=pubkey
      right=10.10.10.30
      rightsubnet=192.168.1.0/24
      rightid="C=CZ, ST=NA, L=Ostrava, O=VSB, OU=MOB, CN=server,
      N=DP, E=tab0012@vsb.cz"
      aaa_identity="C=CZ, ST=NA, L=Ostrava, O=VSB, OU=MOB,
      CN=radius, N=DP, E=tab0012@vsb.cz"
      auto=add
```

/etc/ipsec.secret

```
: PIN %smartcard:6FF32AB18778BCABBB509E597CCAF3E7817E21B
"123456" #%prompt
klient : EAP "12345"
```

/etc/strongswan.d/charon/pkcs11.conf

```
modules {
      pkiclient {
            path = /usr/lib64/libeTPkcs11.so
      }
}
```

/etc/strongswan.conf

```
charon {
        load_modular = yes
        pugins {
                include strongswan.d/charon/*.conf
        }
}
include strongswan.d/*.conf
```

Příloha N: *StrongSwan – výpisy vytváření spojeni a výpis z Wiresharku*

Výpisy ze strany serveru

ipsec restart --nofork

```
Stopping strongSwan IPsec...

Starting strongSwan 5.1.2 IPsec [starter]...

00[DMN] Starting IKE charon daemon (strongSwan 5.1.2, Linux
3.13.0-45-generic, x86_64)

00[CFG] loading ca certificates from '/etc/ipsec.d/cacerts'

00[CFG]   loaded ca certificate "C=CZ, ST=NA, L=Ostrava, O=VSB,
OU=MOB, CN=VSB CA, N=DP, E=tab0012@vsb.cz" from
'/etc/ipsec.d/cacerts/ca.crt'

00[CFG] loading aa certificates from '/etc/ipsec.d/aacerts'

00[CFG] loading ocsp signer certificates from
'/etc/ipsec.d/ocspcerts'

00[CFG] loading attribute certificates from
'/etc/ipsec.d/acerts'

00[CFG] loading crls from '/etc/ipsec.d/crls'

00[CFG] loading secrets from '/etc/ipsec.secrets'

00[CFG]   loaded RSA private key from
'/etc/ipsec.d/private/192.168.1.106.key'

00[LIB] loaded plugins: charon test-vectors pkcs11 aes rc2 sha1
sha2 md4 md5 random nonce x509 revocation constraints pkcs1
pkcs7 pkcs8 pkcs12 pem openssl xcbc cmac hmac ctr ccm gcm attr
kernel-netlink resolve socket-default stroke updown eap-identity
eap-md5 eap-mschapv2 eap-radius eap-tls eap-ttls eap-peap
addrblock

00[LIB] unable to load 5 plugin features (5 due to unmet
dependencies)

00[LIB] dropped capabilities, running as uid 0, gid 0

00[JOB] spawning 16 worker threads

charon (24384) started after 60 ms

06[CFG] received stroke: add connection 'tunel'

06[CFG] adding virtual IP address pool 10.0.0.0/24

06[CFG]   loaded certificate "C=CZ, ST=NA, L=Ostrava, O=VSB,
OU=MOB, CN=192.168.1.106, N=DP, E=tab0012@vsb.cz" from
'server.crt'
```

```
06[CFG] added configuration 'tunel'
^C00[DMN] signal of type SIGINT received. Shutting down
charon stopped after 200 ms
ipsec starter stopped
```

ipsec statusall – neaktivní spojení

```
Status of IKE charon daemon (strongSwan 5.1.2, Linux 3.13.0-45-
generic, x86_64):
  uptime: 6 seconds, since Feb 12 21:16:24 2015
  malloc: sbrk 1351680, mmap 0, used 360848, free 990832
  worker threads: 11 of 16 idle, 5/0/0/0 working, job queue:
0/0/0/0, scheduled: 0
  loaded plugins: charon test-vectors pkcs11 aes rc2 sha1 sha2
md4 md5 random nonce x509 revocation constraints pkcs1 pkcs7
pkcs8 pkcs12 pem openssl xcbc cmac hmac ctr ccm gcm attr kernel-
netlink resolve socket-default stroke updown eap-identity eap-
md5 eap-mschapv2 eap-radius eap-tls eap-ttls eap-peap addrblock
Virtual IP pools (size/online/offline):
  10.0.0.0/24: 254/0/0
Listening IP addresses:
  10.10.10.30
Connections:
       tunel:  10.10.10.30...%any  IKEv2
       tunel:   local:  [C=CZ, ST=NA, L=Ostrava, O=VSB, OU=MOB,
CN=server, N=DP, E=tab0012@vsb.cz] uses public key
authentication
       tunel:    cert:  "C=CZ, ST=NA, L=Ostrava, O=VSB, OU=MOB,
CN=server, N=DP, E=tab0012@vsb.cz"
       tunel:   remote: uses EAP_RADIUS authentication
       tunel:   child:  192.168.1.0/24 === dynamic TUNNEL
Security Associations (0 up, 0 connecting):
  none
```

ipsec statusall – aktivní spojení

```
Status of IKE charon daemon (strongSwan 5.1.2, Linux 3.13.0-45-
generic, x86_64):
```

```
  uptime: 25 minutes, since Feb 12 21:16:24 2015
  malloc: sbrk 1486848, mmap 0, used 391232, free 1095616
  worker threads: 11 of 16 idle, 5/0/0/0 working, job queue:
0/0/0/0, scheduled: 5
  loaded plugins: charon test-vectors pkcs11 aes rc2 sha1 sha2
md4 md5 random nonce x509 revocation constraints pkcs1 pkcs7
pkcs8 pkcs12 pem openssl xcbc cmac hmac ctr ccm gcm attr kernel-
netlink resolve socket-default stroke updown eap-identity eap-
md5 eap-mschapv2 eap-radius eap-tls eap-ttls eap-peap addrblock
Virtual IP pools (size/online/offline):
  10.0.0.0/24: 254/1/0
Listening IP addresses:
  10.10.10.30
Connections:
       tunel:  10.10.10.30...%any  IKEv2
       tunel:   local:  [C=CZ, ST=NA, L=Ostrava, O=VSB, OU=MOB,
CN=server, N=DP, E=tab0012@vsb.cz] uses public key
authentication
       tunel:    cert:  "C=CZ, ST=NA, L=Ostrava, O=VSB, OU=MOB,
CN=server, N=DP, E=tab0012@vsb.cz"
       tunel:   remote: uses EAP_RADIUS authentication
       tunel:   child:  192.168.1.0/24 === dynamic TUNNEL
Security Associations (1 up, 0 connecting):
       tunel[2]: ESTABLISHED 21 seconds ago, 10.10.10.30[C=CZ,
ST=NA, L=Ostrava, O=VSB, OU=MOB, CN=192.168.1.106, N=DP,
E=tab0012@vsb.cz]...20.20.20.10[klient]
       tunel[2]: IKEv2 SPIs: 8a62ea8db0190294_i
9dbc3bb7d4360b87_r*, public key reauthentication in 2 hours
       tunel[2]: IKE proposal:
AES_CBC_256/HMAC_SHA1_96/PRF_HMAC_SHA1/MODP_1024
       tunel{2}:  INSTALLED, TUNNEL, ESP SPIs: cfaa582c_i
c5db408a_o
       tunel{2}:  AES_CBC_256/HMAC_SHA1_96, 0 bytes_i, 0
bytes_o, rekeying in 45 minutes
       tunel{2}:   192.168.1.0/24 === 10.0.0.1/32
```

ipsec status

```
Security Associations (1 up, 0 connecting):
       tunel[2]: ESTABLISHED 18 seconds ago, 10.10.10.30[C=CZ,
ST=NA, L=Ostrava, O=VSB, OU=MOB, CN=server, N=DP,
E=tab0012@vsb.cz]...20.20.20.10[klient]
       tunel{2}:  INSTALLED, TUNNEL, ESP SPIs: cfaa582c_i
c5db408a_o
       tunel{2}:   192.168.1.0/24 === 10.0.0.1/32
```

Výpisy ze strany klienta

ipsec restart --nofork

```
Stopping strongSwan IPsec...
Starting strongSwan 5.1.2 IPsec [starter]...
00[DMN] Starting IKE charon daemon (strongSwan 5.1.2, Linux
3.13.0-32-generic, x86_64)
00[CFG] loaded PKCS#11 v2.1 library 'pkiclient'
(/usr/lib64/libeTPkcs11.so)
00[CFG]   Aladdin Ltd.: eToken PKCS#11 v5.0
00[CFG]   found token in slot 'pkiclient':0 (Aladdin eToken PRO
64 00 00)
00[CFG]     Aladdin (Aladdin Ltd.: eToken)
00[CFG]     loaded trusted cert 'klient'
00[CFG] loading ca certificates from '/etc/ipsec.d/cacerts'
00[CFG]   loaded ca certificate "C=CZ, ST=NA, L=Ostrava, O=VSB,
OU=MOB, CN=VSB CA, N=DP, E=tab0012@vsb.cz" from
'/etc/ipsec.d/cacerts/ca.crt'
00[CFG] loading aa certificates from '/etc/ipsec.d/aacerts'
00[CFG] loading ocsp signer certificates from
'/etc/ipsec.d/ocspcerts'
00[CFG] loading attribute certificates from
'/etc/ipsec.d/acerts'
00[CFG] loading crls from '/etc/ipsec.d/crls'
00[CFG] loading secrets from '/etc/ipsec.secrets'
00[CFG]   loaded EAP secret for klient
```

```
00[LIB] loaded plugins: charon pkcs11 aes des rc2 sha1 sha2 md5
random nonce x509 revocation constraints pubkey pkcs1 pkcs7
pkcs8 pkcs12 pgp dnskey sshkey pem openssl fips-prf gmp agent
xcbc cmac hmac attr kernel-netlink resolve socket-default stroke
updown eap-identity eap-md5 eap-gtc eap-mschapv2 eap-tls eap-
ttls eap-peap xauth-generic

00[LIB] unable to load 3 plugin features (3 due to unmet
dependencies)

00[JOB] spawning 16 worker threads

charon (3055) started after 640 ms

11[CFG] received stroke: add connection 'tunel'

11[CFG]   loaded certificate "C=CZ, ST=NA, L=Ostrava, O=VSB,
OU=MOB, CN=klient, N=DP, E=tab0012@vsb.cz" from
'%smartcard:6FF32AB18778BCABBB509E597CCAF3E7817E201B'

11[CFG] added configuration 'tunel'
```

ipsec statusall – aktivní spojení

```
Status of IKE charon daemon (strongSwan 5.1.2, Linux 3.13.0-32-
generic, x86_64):
  uptime: 12 minutes, since Apr 16 18:43:48 2015
  malloc: sbrk 1486848, mmap 0, used 455536, free 1031312
  worker threads: 10 of 16 idle, 6/0/0/0 working, job queue:
0/0/0/0, scheduled: 4
  loaded plugins: charon pkcs11 aes des rc2 sha1 sha2 md5 random
nonce x509 revocation constraints pubkey pkcs1 pkcs7 pkcs8
pkcs12 pgp dnskey sshkey pem openssl fips-prf gmp agent xcbc
cmac hmac attr kernel-netlink resolve socket-default stroke
updown eap-identity eap-md5 eap-gtc eap-mschapv2 eap-tls eap-
ttls eap-peap xauth-generic
Listening IP addresses:
  20.20.20.10
Connections:
       tunel:  20.20.20.10...10.10.10.30  IKEv2
       tunel:   local:  [klient] uses EAP authentication
       tunel:    cert:  "C=CZ, ST=NA, L=Ostrava, O=VSB, OU=MOB,
CN=klient, N=DP, E=tab0012@vsb.cz"
```

```
       tunel:   remote: [C=CZ, ST=NA, L=Ostrava, O=VSB, OU=MOB,
CN=server, N=DP, E=tab0012@vsb.cz] uses public key
authentication

       tunel:   child:  dynamic === 192.168.1.0/24 TUNNEL

Security Associations (1 up, 0 connecting):

       tunel[2]: ESTABLISHED 3 minutes ago,
20.20.20.10[klient]...10.10.10.30[C=CZ, ST=NA, L=Ostrava, O=VSB,
OU=MOB, CN=server, N=DP, E=tab0012@vsb.cz]

       tunel[2]: IKEv2 SPIs: 8a62ea8db0190294_i*
9dbc3bb7d4360b87_r, EAP reauthentication in 44 minutes

       tunel[2]: IKE proposal:
AES_CBC_256/HMAC_SHA1_96/PRF_HMAC_SHA1/MODP_1024

       tunel{2}:  INSTALLED, TUNNEL, ESP SPIs: c5db408a_i
cfaa582c_o

       tunel{2}:  AES_CBC_256/HMAC_SHA1_96, 0 bytes_i, 0
bytes_o, rekeying in 3 minutes

       tunel{2}:   10.0.0.1/32 === 192.168.1.0/24
```

ipsec up tunel – výpis navázání spojení

```
initiating IKE_SA tunel[1] to 10.10.10.30

generating IKE_SA_INIT request 0 [ SA KE No N(NATD_S_IP)
N(NATD_D_IP) ]

sending packet: from 20.20.20.10[500] to 10.10.10.30[500] (304
bytes)

received packet: from 10.10.10.30[500] to 20.20.20.10[500] (337
bytes)

parsed IKE_SA_INIT response 0 [ SA KE No N(NATD_S_IP)
N(NATD_D_IP) CERTREQ N(MULT_AUTH) ]

received cert request for "C=CZ, ST=NA, L=Ostrava, O=VSB,
OU=MOB, CN=VSB CA, N=DP, E=tab0012@vsb.cz"

sending cert request for "C=CZ, ST=NA, L=Ostrava, O=VSB, OU=MOB,
CN=VSB CA, N=DP, E=tab0012@vsb.cz"

establishing CHILD_SA tunel

generating IKE_AUTH request 1 [ IDi N(INIT_CONTACT) CERTREQ IDr
CPRQ(ADDR DNS) SA TSi TSr N(MOBIKE_SUP) N(NO_ADD_ADDR)
N(MULT_AUTH) N(EAP_ONLY) ]
```

```
sending packet: from 20.20.20.10[4500] to 10.10.10.30[4500] (412
bytes)

received packet: from 10.10.10.30[4500] to 20.20.20.10[4500]
(1772 bytes)

parsed IKE_AUTH response 1 [ IDr CERT AUTH EAP/REQ/PEAP ]

received end entity cert "C=CZ, ST=NA, L=Ostrava, O=VSB, OU=MOB,
CN=server, N=DP, E=tab0012@vsb.cz"

  using certificate "C=CZ, ST=NA, L=Ostrava, O=VSB, OU=MOB,
CN=server, N=DP, E=tab0012@vsb.cz"

  using trusted ca certificate "C=CZ, ST=NA, L=Ostrava, O=VSB,
OU=MOB, CN=VSB CA, N=DP, E=tab0012@vsb.cz"

checking certificate status of "C=CZ, ST=NA, L=Ostrava, O=VSB,
OU=MOB, CN=server, N=DP, E=tab0012@vsb.cz"

certificate status is not available

  reached self-signed root ca with a path length of 0

authentication of 'C=CZ, ST=NA, L=Ostrava, O=VSB, OU=MOB,
CN=server, N=DP, E=tab0012@vsb.cz' with RSA signature successful

server requested EAP_PEAP authentication (id 0x01)

EAP_PEAP version is v0

generating IKE_AUTH request 2 [ EAP/RES/PEAP ]

sending packet: from 20.20.20.10[4500] to 10.10.10.30[4500] (236
bytes)

received packet: from 10.10.10.30[4500] to 20.20.20.10[4500]
(1100 bytes)

parsed IKE_AUTH response 2 [ EAP/REQ/PEAP ]

negotiated TLS 1.0 using suite
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA

generating IKE_AUTH request 3 [ EAP/RES/PEAP ]

sending packet: from 20.20.20.10[4500] to 10.10.10.30[4500] (76
bytes)

received packet: from 10.10.10.30[4500] to 20.20.20.10[4500]
(1100 bytes)

parsed IKE_AUTH response 3 [ EAP/REQ/PEAP ]

generating IKE_AUTH request 4 [ EAP/RES/PEAP ]
```

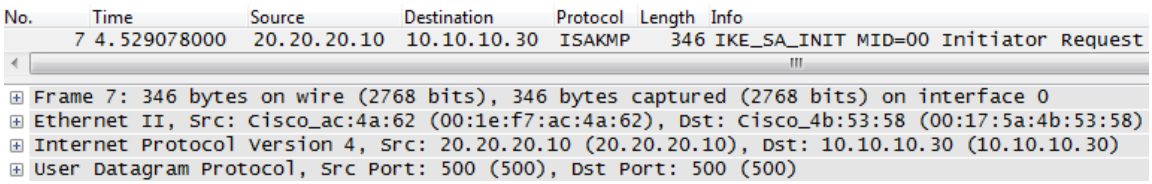
```
sending packet: from 20.20.20.10[4500] to 10.10.10.30[4500] (76
bytes)

received packet: from 10.10.10.30[4500] to 20.20.20.10[4500]
(1020 bytes)

parsed IKE_AUTH response 4 [ EAP/REQ/PEAP ]

received TLS server certificate 'C=CZ, ST=NA, L=Ostrava, O=VSB,
OU=MOB, CN=radiusC, N=DP, E=tab0012@vsb.cz'

received TLS intermediate certificate 'C=CZ, ST=NA, L=Ostrava,
O=VSB, OU=MOB, CN=VSB CA, N=DP, E=tab0012@vsb.cz'

  using certificate "C=CZ, ST=NA, L=Ostrava, O=VSB, OU=MOB,
CN=radiusC, N=DP, E=tab0012@vsb.cz"

  using trusted ca certificate "C=CZ, ST=NA, L=Ostrava, O=VSB,
OU=MOB, CN=VSB CA, N=DP, E=tab0012@vsb.cz"

checking certificate status of "C=CZ, ST=NA, L=Ostrava, O=VSB,
OU=MOB, CN=radiusC, N=DP, E=tab0012@vsb.cz"

certificate status is not available

  reached self-signed root ca with a path length of 0

received TLS cert request for 'C=CZ, ST=NA, L=Ostrava, O=VSB,
OU=MOB, CN=VSB CA, N=DP, E=tab0012@vsb.cz

sending TLS peer certificate 'C=CZ, ST=NA, L=Ostrava, O=VSB,
OU=MOB, CN=klient, N=DP, E=tab0012@vsb.cz'

generating IKE_AUTH request 5 [ EAP/RES/PEAP ]

sending packet: from 20.20.20.10[4500] to 10.10.10.30[4500]
(1100 bytes)

received packet: from 10.10.10.30[4500] to 20.20.20.10[4500] (76
bytes)

parsed IKE_AUTH response 5 [ EAP/REQ/PEAP ]

generating IKE_AUTH request 6 [ EAP/RES/PEAP ]

sending packet: from 20.20.20.10[4500] to 10.10.10.30[4500] (716
bytes)

received packet: from 10.10.10.30[4500] to 20.20.20.10[4500]
(140 bytes)

parsed IKE_AUTH response 6 [ EAP/REQ/PEAP ]

generating IKE_AUTH request 7 [ EAP/RES/PEAP ]
```

```
sending packet: from 20.20.20.10[4500] to 10.10.10.30[4500] (76
bytes)

received packet: from 10.10.10.30[4500] to 20.20.20.10[4500]
(108 bytes)

parsed IKE_AUTH response 7 [ EAP/REQ/PEAP ]

received tunneled EAP-PEAP AVP [EAP/REQ/ID]

server requested EAP_IDENTITY authentication (id 0x07)

sending tunneled EAP-PEAP AVP [EAP/RES/ID]

generating IKE_AUTH request 8 [ EAP/RES/PEAP ]

sending packet: from 20.20.20.10[4500] to 10.10.10.30[4500] (108
bytes)

received packet: from 10.10.10.30[4500] to 20.20.20.10[4500]
(140 bytes)

parsed IKE_AUTH response 8 [ EAP/REQ/PEAP ]

received tunneled EAP-PEAP AVP [EAP/REQ/MSCHAPV2]

server requested EAP_MSCHAPV2 authentication (id 0x08)

sending tunneled EAP-PEAP AVP [EAP/RES/MSCHAPV2]

generating IKE_AUTH request 9 [ EAP/RES/PEAP ]

sending packet: from 20.20.20.10[4500] to 10.10.10.30[4500] (172
bytes)

received packet: from 10.10.10.30[4500] to 20.20.20.10[4500]
(156 bytes)

parsed IKE_AUTH response 9 [ EAP/REQ/PEAP ]

received tunneled EAP-PEAP AVP [EAP/REQ/MSCHAPV2]

EAP-MS-CHAPv2 succeeded: '(null)'

sending tunneled EAP-PEAP AVP [EAP/RES/MSCHAPV2]

generating IKE_AUTH request 10 [ EAP/RES/PEAP ]

sending packet: from 20.20.20.10[4500] to 10.10.10.30[4500] (108
bytes)

received packet: from 10.10.10.30[4500] to 20.20.20.10[4500]
(108 bytes)

parsed IKE_AUTH response 10 [ EAP/REQ/PEAP ]

received tunneled EAP-PEAP AVP [EAP/SUCC]

sending tunneled EAP-PEAP AVP [EAP/SUCC]
```

```
generating IKE_AUTH request 11 [ EAP/RES/PEAP ]
sending packet: from 20.20.20.10[4500] to 10.10.10.30[4500] (124
bytes)
received packet: from 10.10.10.30[4500] to 20.20.20.10[4500] (76
bytes)
parsed IKE_AUTH response 11 [ EAP/SUCC ]
EAP method EAP_PEAP succeeded, MSK established
authentication of 'klient' (myself) with EAP
generating IKE_AUTH request 12 [ AUTH ]
sending packet: from 20.20.20.10[4500] to 10.10.10.30[4500] (92
bytes)
received packet: from 10.10.10.30[4500] to 20.20.20.10[4500]
(236 bytes)
parsed IKE_AUTH response 12 [ AUTH CPRP(ADDR) SA TSi TSr
N(AUTH_LFT) N(MOBIKE_SUP) N(ADD_4_ADDR) ]
authentication of 'C=CZ, ST=NA, L=Ostrava, O=VSB, OU=MOB,
CN=server, N=DP, E=tab0012@vsb.cz' with EAP successful
IKE_SA tunel[1] established between
20.20.20.10[klient]...10.10.10.30[C=CZ, ST=NA, L=Ostrava, O=VSB,
OU=MOB, CN=server, N=DP, E=tab0012@vsb.cz]
scheduling reauthentication in 2881s
maximum IKE_SA lifetime 3421s
installing new virtual IP 10.0.0.1
CHILD_SA tunel{1} established with SPIs c57aee40_i c8f717df_o
and TS 10.0.0.1/32 === 192.168.1.0/24
connection 'tunel' established successfully
```

Výpis z Wiresharku

Níže zobrazena zpráva IKE_SA_INIT, odeslána klientem 20.20.20.10.



```
⊟ Internet Security Association and Key Management Protocol
    Initiator SPI: 66937820b59ab9fb
    Responder SPI: 0000000000000000
    Next payload: Security Association (33)
  ⊞ Version: 2.0
    Exchange type: IKE_SA_INIT (34)
  ⊞ Flags: 0x08 (Initiator, No higher version, Request)
    Message ID: 0x00000000
    Length: 304
  ⊟ Type Payload: Security Association (33)
      Next payload: Key Exchange (34)
      0... .... = Critical Bit: Not Critical
      Payload length: 48
    ⊟ Type Payload: Proposal (2) # 1
        Next payload: NONE / No Next Payload  (0)
        0... .... = Critical Bit: Not Critical
        Payload length: 44
        Proposal number: 1
        Protocol ID: IKE (1)
        SPI Size: 0
        Proposal transforms: 4
      ⊟ Type Payload: Transform (3)
          Next payload: Transform (3)
          0... .... = Critical Bit: Not Critical
          Payload length: 12
          Transform Type: Encryption Algorithm (ENCR) (1)
          Transform ID (ENCR): ENCR_AES_CBC (12)
        ⊞ Transform IKE2 Attribute Type (t=14,l=2) Key-Length : 256
      ⊟ Type Payload: Transform (3)
          Next payload: Transform (3)
          0... .... = Critical Bit: Not Critical
          Payload length: 8
          Transform Type: Integrity Algorithm (INTEG) (3)
          Transform ID (INTEG): AUTH_HMAC_SHA1_96 (2)
      ⊟ Type Payload: Transform (3)
          Next payload: Transform (3)
          0... .... = Critical Bit: Not Critical
          Payload length: 8
          Transform Type: Pseudo-random Function (PRF) (2)

      ⊟ Type Payload: Transform (3)
          Next payload: NONE / No Next Payload  (0)
          0... .... = Critical Bit: Not Critical
          Payload length: 8
          Transform Type: Diffie-Hellman Group (D-H) (4)
          Transform ID (D-H): Alternate 1024-bit MODP group (2)
  ⊞ Type Payload: Key Exchange (34)
  ⊞ Type Payload: Nonce (40)
  ⊞ Type Payload: Notify (41)
  ⊞ Type Payload: Notify (41)
```

Příloha O: *Konfigurace serveru strongSwan pro Windows 7 klienty*

```
# ipsec.conf – konfigurace metody EAP-TLS
conn %default
    keyexchange=ikev2
    ike=aes256-sha1-modp1024!
    esp=aes256-sha1!
    dpdaction=clear
    dpddelay=300s
    rekey=no

conn win7
     leftcert=aCert.pem
     leftauth=pubkey
     leftsubnet=192.168.0.0/24
     right=%any
     rightauth=eap-tls
     rightsendcert=never
     rightsourceip=192.168.1.0/24
     eap_identity=%any
     keyexchange=ikev2
     auto=add
```

Důležité aby zde byl parametr `eap_identity`, dále také nastavení Diffie-Hellman skupiny pro úspěšné navázání šlo nejvýše na DH skupinu 2 (modp1024).

Příloha P: *Kompatibilita mezi strongSwan a Cisco 2801*

Cisco IOS konfigurace

```
crypto isakmp policy 10
 hash sha
 encr aes
 authentication pre-share
 group 5
crypto isakmp key cisco address 172.16.10.2

crypto ipsec transform-set TS esp-aes esp-sha-hmac
 mode tunnel

crypto map cmap 10 ipsec-isakmp
 set peer 172.16.10.2
 set transform-set TS
 match address cryptoacl

interface Ethernet0/1
 ip address 192.168.1.1 255.255.255.0

interface Ethernet0/0
 ip address 172.16.10.1 255.255.255.0
 crypto map cmap

ip access-list extended cryptoacl
 permit ip 192.168.1.0 0.0.0.255 192.168.2.0 0.0.0.255
```

strongSwan konfigurace

```
/etc/ipsec.conf
config setup
        # strictcrlpolicy=yes
        # uniqueids = no

conn %default
        ikelifetime=1440m
        keylife=60m
        rekeymargin=3m
        keyingtries=1
        keyexchange=ikev1
        authby=secret
```

```
conn ciscoios
        left=172.16.10.2
        leftsubnet=192.168.2.0/24
        leftid=172.16.10.2
        leftfirewall=yes
        right=172.16.10.1
        rightsubnet=192.168.1.0/24
        rightid=172.16.10.1
        auto=add
        ike=aes128-sha1-modp1536
        esp=aes128-sha1


/etc/ipsec.secrets
172.16.10.2 172.16.10.1 : PSK cisco
```

Router#show crypto session detail

```
Crypto session current status
Code: C - IKE Configuration mode, D - Dead Peer Detection
K - Keepalives, N - NAT-traversal, T - cTCP encapsulation
X - IKE Extended Authentication, F - IKE Fragmentation

Interface: FastEthernet0/0
Uptime: 00:01:19
Session status: UP-ACTIVE
Peer: 172.16.10.2 port 500 fvrf: (none) ivrf: (none)
      Phase1_id: 172.16.10.2
      Desc: (none)
  IKE SA: local 172.16.10.1/500 remote 172.16.10.2/500 Active
         Capabilities:(none) connid:1006 lifetime:23:58:40
  IPSEC FLOW: permit ip 192.168.2.0/255.255.255.0
192.168.1.0/255.255.255.0
        Active SAs: 0, origin: crypto map
        Inbound:  #pkts dec'ed 0 drop 0 life (KB/Sec) 0/0
        Outbound: #pkts enc'ed 0 drop 0 life (KB/Sec) 0/0
```

```
  IPSEC FLOW: permit ip 192.168.1.0/255.255.255.0
192.168.2.0/255.255.255.0
        Active SAs: 2, origin: crypto map
        Inbound:  #pkts dec'ed 0 drop 0 life (KB/Sec)
4501925/3520
        Outbound: #pkts enc'ed 0 drop 0 life (KB/Sec)
4501928/3520


Router#show crypto isakmp sa detail
Codes: C - IKE configuration mode, D - Dead Peer Detection
       K - Keepalives, N - NAT-traversal
       T - cTCP encapsulation, X - IKE Extended Authentication
       psk - Preshared key, rsig - RSA signature
       renc - RSA encryption
IPv4 Crypto ISAKMP SA

C-id  Local Remote I-VRF  Status Encr Hash Auth DH Lifetime Cap.
1006  172.16.10.1 172.16.10.2  ACTIVE aes  sha  psk  5  23:58:24
       Engine-id:Conn-id =  SW:6
```

ipsec statusall – výpis ze strongSwan

```
Status of IKE charon daemon (strongSwan 5.1.2, Linux 3.13.0-46-
generic, x86_64):
  uptime: 6 minutes, since Apr 07 16:02:08 2015
  malloc: sbrk 2568192, mmap 0, used 350960, free 2217232
  worker threads: 11 of 16 idle, 5/0/0/0 working, job queue:
0/0/0/0, scheduled: 6
  loaded plugins: charon test-vectors aes rc2 sha1 sha2 md4 md5
random nonce x509 revocation constraints pkcs1 pkcs7 pkcs8
pkcs12 pem openssl xcbc cmac hmac ctr ccm gcm attr kernel-
netlink resolve socket-default stroke updown eap-identity
addrblock
Listening IP addresses:
  172.16.10.2
  192.168.2.1
```

```
Connections:
    ciscoios:  172.16.10.2...172.16.10.1  IKEv1
    ciscoios:   local:  [172.16.10.2] uses pre-shared key
authentication
    ciscoios:   remote: [172.16.10.1] uses pre-shared key
authentication
    ciscoios:   child:  192.168.2.0/24 === 192.168.1.0/24 TUNNEL
Security Associations (1 up, 0 connecting):
    ciscoios[3]: ESTABLISHED 3 minutes ago,
172.16.10.2[172.16.10.2]...172.16.10.1[172.16.10.1]
    ciscoios[3]: IKEv1 SPIs: 32c915a4a4788dac_i*
d73d8d743521f38b_r, pre-shared key reauthentication in 23 hours
    ciscoios[3]: IKE proposal:
AES_CBC_128/HMAC_SHA1_96/PRF_HMAC_SHA1/MODP_1536
    ciscoios{3}:  INSTALLED, TUNNEL, ESP SPIs: c72a9635_i
666068c4_o
    ciscoios{3}:  AES_CBC_128/HMAC_SHA1_96, 0 bytes_i, 1260
bytes_o (15 pkts, 184s ago), rekeying in 51 minutes
    ciscoios{3}:   192.168.2.0/24 === 192.168.1.0/24
```